Genetic Network Completion Using Dynamic Programming and Least-Squares Fitting(Dissertation_全文)

AUTHOR(S):

Nakajima, Natsu

Genetic Network Completion Using Dynamic Programming and Least-Squares Fitting

動的計画法と最小二乗法を用いた
遺伝子ネットワーク補完

Natsu Nakajima
仲嶋　なつ

Thesis supervisor : Professor Tatsuya Akutsu

Department of Intelligence Science and Technology
Graduate School of Informatics
Kyoto University

Abstract

Genes are essential for the formation and the maintenance of living systems. Genetic systems are controlled by forming extremely complex networks of interactions between multiple genes and other genetic components. The inference of gene regulatory networks has emerged as a general approach to understand how genes regulate one another and there have been many attempts to describe genetic networks using several mathematical models based on microarray observations. In most cases, these approaches are likely to lead to an intrinsic difficulty that the inference of complex interactions involving many genes from only sparse and noisy data may be unrealistic and unreasonable. This thesis presents a new approach, network completion, for inferring genetic networks and considers three network completion problems from different points of view. The first problem is network completion from time-series microarray data, the second problem is directed toward completing time-varying genetic networks from time-series data, and third problem is focusing on completing networks under stationary conditions from static data.

The first half of this thesis describes a novel method, DPLSQ, for the network completion defined as to make the minimum amount of modifications (additions and deletions of edges) to a given genetic network in order to be consistent with the microarray observed data. We assume that the causal relationships between genes are modeled by differential equations. By introducing a least-squares fitting, this problem can be reduced to an optimization problem and solved efficiently using dynamic programming. This combination of least-squares fitting and dynamic programming commonly serves as a basis for the proposed methods in this thesis. The notable feature of DPLSQ is to be able to provide optimality guarantees with polynomial time complexity subject to the constraint on the maximum indegree. The results show that DPLSQ has better performance in terms of the modified edges than existing methods.

In the next part, we present two types of novel methods DPLSQ-TV and DPLSQ-HS that extend DPLSQ for time-varying genetic networks. DPLSQ-TV is an exact method to detect change time points and required modifications of time-varying networks, with a newly introduced double dynamic programming. While DPLSQ-TV can provide optimality guarantees, it has relatively high computational complexity. To overcome this inefficiency, we also present a heuristic method DPLSQ-HS, which can reduce the total computation time by imposing an upper bound on the number of combinations

of incoming nodes. The performance of two methods is evaluated in terms of the time point error for change points, the accuracy of modified edges and the CPU time using synthetic and microarray data. The results show that DPLSQ-TV can provide optimal solutions and relatively better accuracy than an existing method. Although DPLSQ-HS can only provide near-optimal performance with regard to the time point error and the accuracy, it achieves significant performance improvement on computational complexity.

In the last part, we propose a new method named as DPLSQ-SS, for solving the problem to complete and infer genetic network architecture under stationary conditions based on static microarray data. This problem has a twofold purpose: (1) to complete and reconstruct static gene networks from static expression profile, and (2) to investigate the relationships between genes under different stationary conditions. Based on the main idea of DPLSQ, DPLSQ-SS is developed by modifying an objective function to deal with static expression data. As the results of experiments to assess the accuracy of modified edges and CPU time, DPLSQ-SS provides relatively good inference accuracy in comparison to existing methods. Additionally, we conduct an experiment with two-types of microarray data taken from lung cancer and normal lung samples. This result suggests that DPLSQ-SS can identify significant differences between the cancer and the normal static networks. Therefore, it can be expected that DPLSQ-SS will be widely applicable to examine the regulatory relationships among genes expressed under various stationary conditions.

Acknowledgments

First of all, I would like to express my deepest appreciation to my supervisor, Professor Tatsuya Akutsu for giving the opportunity to carry out my doctoral research and for their continuous support. I am indebted to him for his constant assistance, excellent guidance and sound advice in my research and academic life.

I also would like to express my sincere gratitude to my co-supervisors, Professor Akihiro Yamamoto and Professor Yasuo Okabe. Their insightful and thoughtful suggestions were truly invaluable and gave me the opportunity to rediscover the interests for this research. Without their kindness and generosity, I would not have been able to accomplish my thesis.

I would like to extend my deepest thank to my advisor, Associate Professor Jesper Jansson for all his help not just academic study but also campus life. Especially, I offer my thanks to him for valuable comments towards completing this thesis.

Additionally, I would like to many thank Professor Hideo Matsuda in Osaka University who provided the useful discussions and inspired me to learn a new subject. My special thanks also go to Dr. Yoshihiro Yamanishi in Kyushu University and Dr. Katsuhisa Horimoto in CBRC for their collaboration with my first paper.

I am immensely grateful for my friends Mr. Kenichi Urai for his devoted support and continued encouragement in my life and Mr. Yuki Narita who has provided me with tremendous knowledge and useful discussions.

Finally, I wish to acknowledge each and everyone who has assisted and supported me along the way through both my academic and personal life over the years.
Thank you.

Natsu Nakajima
December 2014

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Genetic Network Inference

All living things are composed of one or more cells which are called the "basic building blocks of life". In particular, the human body is said to be made of over trillion of cells. The cell can control its environment by proteins that carry out many different roles, so proteins are essential molecules and vital for life. However, in fact, proteins are built from information originating from genes. The gene is the basic unit of genetic information stored in DNA sequences composed of double strand in every cell. Gene expression is considered a common phenomenon to all living things and whether a gene is expressed or not is to make an effect on biological systems. Because the condition of biological system is flexibly controlled by altering different gene expression patterns induced by complex interactions between genetic, epigenetic and environmental factors in response to internal and external cues.

Gene expression is the process by which genetic information encoded in DNA is converted into proteins and is crucial for the maintenance and the development of living systems. "How is gene expression controlled?" – The answer to this question is founded by Jacob and Monod in 1961 [1]. Their discovery revealed that the existence of messenger RNA that carries the genetic information from DNA and the operon theory for gene regulation. The principle of this theory is that genes can act to control their own expression, which exhibit switch-like behavior between the two states; "ON (expressed)" and "OFF (not expressed)" only when they are needed, in response to internal and external environments. In other words, even if certain genes are normally supposed to be turned "OFF", they get switched "ON" only under specific environmental conditions, affected by interactions with other genes. This means that whether a gene is expressed or not should be highly regulated not only by joint effects of multiple genes but also by complex interactions caused by both genetic and environmental factors. These complicated interactions between genes seem likely to form "interaction networks" and have become recognized as so-called gene regulatory networks (GRNs).

Recent advances in molecular biological techniques will enable us to gain in-depth understanding of genetic phenomena and at the same time they provide the strong motivation of rebuilding the behavior of them in silico by means of simulation. In the mid-1990s, DNA microarray technology has revolutionized genetics and molecular biology, and of course, it has brought the rapid expansion into the field of systems biology. Microarray allows the simultaneous measurements of RNA expression levels for thousands of genes in a single experiment, thereby providing multiple types of profiles for distinct expression patterns through repeated experiments conducted under various conditions (refer to Section 2.1.2 for further details). This obviously has been helpful for achieving an important goal of systems biology of computational designing and building GRNs based on information derived from gene expression data. Nowadays, these statistical and quantitative approaches for genetic systems become known as genetic network inference or reverse-engineering.

Many in silico methods have been proposed to reconstruct or infer GRNs from expression profiles and most of them typically are developed to build the architecture of GRNs which are modelled using mathematical concepts, taking into consideration the temporal characteristics of expression profiles such as time-series or static. In general, mathematical models to describe GRNs can be expressed in terms of variables representing the expression measurements of genes and regulatory relationships between two variables. These models will be roughly classified into two types: discrete and continuous models according to the quantified measurements. A pioneering study of GRNs has been conducted by Kauffman in 1969 [2], in which the Boolean network was proposed as the simplest discrete model to describe the dynamics of GRNs. A Boolean network consists of a set of nodes representing genes that can take only on the binary values 0 or 1 and a set of edges reflect the causal relationships between genes defined by a list of Boolean functions. In Kauffman networks, since each node is randomly assigned one of the Boolean functions generated from all possible combination of input nodes, the expression state at the next step is determined by the states of its parent nodes at the previous time step [3]. This study would appear to provide evidence that genes indeed interact with one another in the form of the complicated regulatory network that exhibits switch-like behavior and the Boolean networks are suitable models that can describe the key property of genetic regulations more simply.

While Boolean network models are suitable approximations to be able to deal with both time-series and static data, the binary discretization of expression measurements is likely to induce the information loss. Moreover, since the dynamics of its system is updated synchronously at discrete time steps, Boolean networks do exhibit deterministic behavior [4]. To overcome this weakness, a revised model, the probabilistic Boolean network (PBN) has been proposed in [5, 6]. A PBN is known as a stochastic model, where multiple Boolean functions are assigned to each node and one of them can be determined with the corresponding selection probabilities at each time step.

PBNs might indeed reflect the inherent stochasticity in genetic systems.

Another common discrete stochastic model is a Bayesian network (BN) defined as a directed acyclic graph [7, 8, 9, 10]. It describes conditional dependencies among variables, where the state of each variable depends on a subset of other variables and the probability of one event is conditional on that of a previous one using the chain rule. This means that the joint probability can be written as the product of a marginal probability and a conditional probability. Thus, in BNs, arrows indicate not only directions of regulatory relationships but also conditional dependencies. The BN model appears to be effective for reverse-engineering, capable of handling the inherent stochastic aspects of gene expression and noisy measurements [11], but this model has one drawback, because of its acyclicity constraint which means that feedback loops are not allowed [12]. Additionally, the temporal evolution of gene expression cannot be captured by the BNs model. Fortunately, these limitations may be overcome by developing dynamic Bayesian networks (DBNs) [13, 14, 15], which is an extension of BNs to describe how the system will evolve against time. Unlike BNs, the acyclicity constraint is relaxed by considering the temporal property of gene expression so that each variable can freely select its candidate parent variables from all other variables [16]. However, it must be noted that these types of methods based on the probabilistic concepts will commonly require much computational time in case that a large number of genes are involved [17].

In contrast to discrete models, continuous models can describe the genetic regulations with continuous variables, without discretizing the expression measurements. Most common continuous models are able to describe the system dynamics of GRNs more directly and accurately defined by differential equations, which include linear ordinary differential equations (ODEs) [18, 19, 20, 21], nonlinear power-law differential equations [22, 23], kinetic equations [24] and so on. The main objective of applying differential equation models is to effectively identify the network architecture and estimate unknown model parameters based on the expression data.

Generally, most of ODEs models quantify the change rate in the expression level of a particular gene as the derivative function with respect to the expression levels of all of the other related genes at a previous time. In fact, once the model function is determined, estimation of suitable parameters may often be considered as solving a large-scale parameter optimization problem. For instance, the modelling of the stable behavior in [21] reduced the linear ODEs to a form of linear regression to solve the equations. This means that the differential equation based model is suitable to describe the complex behavior underlying genetic phenomena by using continuous measurement values. Notably, this model also enables us to characterize not only positive/negative genetic correlations but also feedback effects such as up/down regulations between genes or self-regulations more naturally and automatically [25]. Therefore, unlike the discrete modeling, differential equation models are quantitative and flexible, capable of

capturing the theoretical aspects behind gene expression [26]. However, it must also be noted that the behavior of systems is usually supposed to depend on model parameters, thus it will be necessary to design in such a way as to avoid the regression overfitting of the data.

There exist other kinds of inference methods based on the concept of information theory as represented by mutual information (MI). MI provides a general measurement to evaluate dependencies between two random variables. For example, ARACNE [27, 28] applies MI to recovering genetic regulations. It identifies the statistically significant dependencies between two genes with removing the vast majority of indirect candidate interactions between genes. However, ARACNE has a serious drawback that it provides only no directional dependencies, so causal relations between genes remain unclear. Information theory based models are simple and have low computational costs, thus they are suitable for large-scale networks. However, these models are unable to handle the complex interactions among multiple genes [29].

Regardless of the variable properties, various approaches for inference of GRNs are also proposed. In particular, a novel framework is established in [30], for integration microarray profile with other biological information such as protein-protein interaction data, protein-DNA data and literature with a BN model. Another study in [31] introduced a similar method which describes dynamics of RNA transcriptional regulations taking into account the location information.

In this way, there has been several approaches to infer the dynamics of genetic regulatory systems based on the core information extracted from microarray experiments. In the following sections, we will introduce a new approach to reverse-engineer GRNs, network completion, and it is the theme of this thesis.

1.2 Genetic Network Completion

Inference or reverse-engineering of GRNs must indeed have been an effective recovery approach to investigate and explain complex phenomena of genetic regulations using genome wide expression profiling in the field of systems biology. Technological innovation in experimental biology has provided advanced and valuable knowledge, that is to say, DNA microarrays can provide comprehensive “snapshots” of dynamic patterns of gene expression that underlie in biological processes and could be rich information about various aspects of genetic regulation. At the same time, there is still not enough detailed information to faithfully reconstruct GRNs that the actual gene regulation appears to exhibit. The reason for this is that firstly, microarray experiments are still quite expensive to perform, so in reality it is hard to obtain a large number of measurements. Secondly, microarray time-series data typically consist of few sampled time points in time (the actual number of sampled time points rarely exceeds a few dozen), while reverse-engineering methods require a relatively large number of sampled points

(refer to details in Section 2.1.2). Thus, these inferences may not look realistic enough for describing the actual phenomena in gene expression. In other words, the inference of complex interactions involving many genes from only sparse and noisy data obviously leads to intrinsic difficulty and poor reconstruction [32]. In addition to this, existing computational methods for network inference, in many cases, require large amounts of computation time for estimating parameters, thereby suffering from computational inefficiencies and loss of accuracy. Hence, in fact, the gold standard method using only expression data has not yet been established. Therefore, a more realistic and reasonable approach is needed to faithfully describe actual gene regulatory dynamics.

In this thesis, we will focus our attention on a new approach for analysis of GRNs, known as "network completion". In recent years, there have been several studies for network completion, not necessarily for biological networks but also for social networks and web graphs. The basic principle of network completion is that a certain type of existing prototype network is given, a completed network can be obtained by using existing knowledge. For instance, applying the social and information networks, Kim *et al.* [33] addressed the network completion problem whose objective is to detect the unobserved parts that should be made up, given an incomplete network including unobserved nodes and edges. They proposed KronEM, which combined the Expectation Maximization with the Kronecker graphs model to detect the missing part of the given network. Similarly, the study by [34] defined the network completion as a problem of inferring the rest of the network structures, when an observed incomplete network is given, and proposed a sampling method to derive confidence intervals from sample networks. Another method for applying to the actual biological data was presented by [35] to be able to reconstruct both missing and spurious interactions in known complex biological networks underlying protein interactions, by using the stochastic block model to capture the structural features in the networks. As a related work, Saito *et al.* [36] developed a method to measure the consistency of an inferred network with the observed gene expression data.

Independently, a different type of network completion model is introduced by Akutsu *et al.* [37] by following the principle of scientific discovery, Occam's razor. The purpose of the study is to make the minimum amount of modifications to an initial network, given a well-known network and an observed dataset, so that the resulting network (completed network) is most consistent with the observed data. It was directed toward inferring of signaling pathway networks composed of a series of proteins underlying the cellular processes and they introduced the Boolean network based method. While this method could allow the regulatory flexibility of biological systems by taking into account the intrinsic or extrinsic types of noise, it could not handle the addition of edges because of its high computational complexity.

Based on this idea, the thesis will provide more practical methods for network completion of GRNs from gene expression microarray data. The novelty of proposed

methods commonly lies in the combination of least-squares fitting to estimate parameters and dynamic programming to automatically detect the architecture of GRNs. The following chapters will present three problems for network completion from different points of view and will describe the details of three novel methods to solve these problems. Since the origin of the word 'complete' is to fill up all the appropriate or necessary parts [38], the network completion should correspond to additions of edges only. In this thesis, however, we will handle the modification operations involving additions and deletions of edges under the term of completion. These researches will be expected to be new initiative for genetic network analysis in the field of bioinformatics and systems biology.

1.3 Scope and Contribution of This Thesis

The goal of this thesis is to minimally modify a given well-known network so as to be consistent with microarray observations. To achieve this aim, we develop novel computational methods for completing and inferring gene regulatory networks using microarray expression data.

In Chapter 3, we begin with network completion that is a new reverse-engineering approach for GRNs from time-series gene expression data. We study various existing computational tools for genetic network inference and examine the fundamental weakness of existing applications. We find out that they do not appear to be realistic and reasonable to rebuild genetic networks based only on information derived from gene expression data. Therefore, we introduce a new approach, network completion for GRNs and propose a novel method named DPLSQ, based on the combination of least-squares fitting and dynamic programming. We aim at minimizing the objective function denoted by the sum-of-squared residuals between the observed and predicted values when adding and/or deleting edges to a node. Our method has a notable feature that guarantees to the optimality of its solution in polynomial time subject to the constraint that the number of maximum indegree of an input network is bounded by a constant. The experimental results show that DPLSQ outperforms existing methods in terms of accuracy of modified edges. In particular, it maintains high precision, regardless of the size of the network. Therefore, the proposed method will be likely to be a new type of method to reconstruct GRNs from time-series expression data.

In Chapter 4, we devote to the problem of network completion for time-varying genetic networks. Biological systems commonly exhibit complex and rich dynamical behaviors affected by intrinsic and extrinsic factors. In particular, it is clear that these properties are caused by structural alterations of gene regulatory networks over time. Therefore, we present two novel network completion methods DPLSQ-TV and DPLSQ-HS that extend DPLSQ for time-varying networks, whose objectives are to detect change time points and structural changes of a given network at specified time intervals.

The novelty of two methods is to introduce a double dynamic programming algorithm where the inner and the outer loops take detecting structural changes and change points, respectively. DPLSQ-TV provides an exact solution in polynomial time, but it has a high computational complexity if the large size of the network is given. Thus, we also propose a heuristic method DPLSQ-HS, to improve the computational efficiency of DPLSQ-TV. The results from computational experiments reveal that both methods can precisely detect change points and the modified edges relatively accurately. DPLSQ-TV can achieve optimal performance and may be suitable for wide range of applications in network completion as well as network inference. DPLSQ-HS, in many cases, can be expected to provide near-optimal performance in terms of detection of change points with much reduced time complexity.

In Chapter 5, we consider the problem of network completion for genetic networks under stationary conditions from static microarray data. A time-course microarray indeed provides valuable information about temporal dynamics underlying GRNs. Similarly, static data might be helpful to analyze the genetic networks if the conditions are a disease, such as cancer, and normal. Thus, we introduce a novel method for network completion DPLSQ-SS, to describe the static behavior of genetic networks and evaluate differences between distinct networks under different stationary conditions. Unlike the above methods, we adopt a new formulation to model the stationary state of gene regulations in terms of nonlinear equations. The basic idea of DPLSQ is also incorporated into DPLSQ-SS, thereby providing optimal solutions in polynomial time subject to some constraints. For the experiments with synthetic data, DPLSQ-SS show relatively good performance compared to other methods. We also conduct the comparative analysis to investigate how different is the cancer network from the normal network predicted from the cancer and healthy samples. The result indicates that the proposed method indeed enables us to detect the structural differences between two types of static networks using static expression data.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 provides the preliminary knowledge consisting of two parts: biological knowledge and the mathematical optimization approach. At the former part of this chapter, we will introduce the concept of gene regulatory networks, and also provide an overview of DNA microarray technology. In the latter part, after describing a formal definition of mathematical optimization problems, we will present general approaches to solve these problems, giving a relevant example.

Chapter 3 describes the proposed method for network completion defined as an optimization problem. After we briefly review of the background and related works in genetic network inference, we present a new method named DPLSQ, based on the

differential equation model and explain DPLSQ in more detail. To evaluate the potential effectiveness of DPLSQ, we conduct computational experiments and compare our performance with existing methods using artificially generated data. We also perform a test on inference of an actual genetic network from real microarray data. At the end of this chapter, we discuss and conclude the validity of the proposed method.

Chapter 4 is dedicated to the problem of network completion for time-varying genetic networks from time-series data and presents two novel methods DPLSQ-TV and DPLSQ-HS that extend DPLSQ for time-varying networks. After explaining the background to this problem, we provide the detailed description of DPLSQ-TV, which is developed for solving the change point detection problem to identify the change time points at which the network topology may change by applying a novel double dynamic programming algorithm. Moreover, to overcome the computational inefficiency of DPLSQ-TV, we propose a heuristic method DPLSQ-HS, which can reduce the time complexity by imposing the constraints. The performance tests are carried out by comparing with those of existing methods. At the end of this chapter, we show that two methods can provide an optimal or even near-optimal solution using synthetic and microarray data. Finally, we conclude this chapter by discussing on the characteristics and the possibilities of two methods.

Chapter 5 is devoted to a new problem of network completion to identify the network structures under stationary conditions from static microarray observations. At the beginning, we explain the motivation behind this research. Then, we explain how to express the static behavior of genetic regulations in terms of nonlinear equations and show that this problem can also be reduced to an optimization problem using the same basic idea of DPLSQ. At the same time, we give a detailed explanation of DPLSQ-SS for solving this problem. Through the computational experiments, we came to the conclusion that DPLSQ-SS can accurately detect the static networks of genetic interactions and can find the structural difference between two types of static networks in the normal state and the cancer state. In the end, we conclude with the brief discussion on the advantages of DPLSQ-SS.

Chapter 6 provides the overall conclusions and future directions.

Chapter 2

Preliminaries

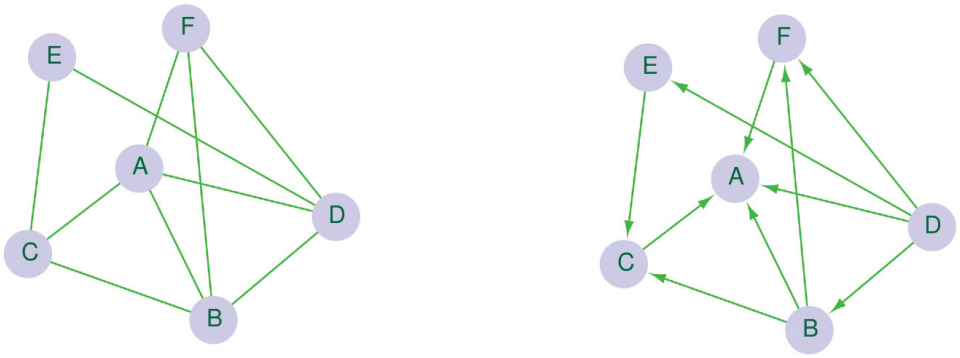
This chapter provides preliminary knowledge and is divided into two main parts. The first part describes prior knowledge into the inference of gene regulatory networks. In particular, we will introduce the basic concept of gene regulatory networks and DNA microarrays. The second part gives an overview of mathematical optimization. After a brief introduction to optimization problems, we will explain them in more detail with a relevant example. Moreover, two strategies to reach our goal will also be provided: least-squares fitting and dynamic programming techniques.

2.1 Genetic Regulatory Network

Genes contain the genetic information stored in DNA whose sequences form the double strand structure by complementary base-pairing. This information is in turn transcribed into messenger RNA (mRNA) in the form of single strand. After the transcription, RNA sequences are translated into amino acid sequences of proteins, which have crucial roles in biological systems. In this way, genes carry genetic information needed to make proteins through the process of transcription and translation. This means that gene expression is the process of converting genetic information into protein synthesis.

In gene expression, it has been clear that genes exhibit switch like behavior between two expression states [1]; "ON" (expressed) or "OFF" (not expressed) affected by the complicated interaction of other genes or multiple proteins in such a way as to form "gene regulatory networks". Thus, gene regulatory networks (GRNs) appear to reflect snapshots of gene expression that might underlie several biological processes (e.g., metabolic, stress and immune responses, disease progression) and under a variety of conditions, such as distinct phases of cell-cycle or life-cycle processes.

GRNs are typically represented by graphs, whose nodes reflect genes and edges imply regulatory connections between two genes. In GRNs, the directed edges indicate that directions of causal interactions between any two genes. Unlike directed graphs, in the case of undirected graphs, the edges reflect only correlations or dependency



(a) Undirected graph (b) Directed graph

Figure 2.1: Gene regulatory networks usually can be modeled by two types of graphs; the directed graph and the undirected graph. The relationship between two genes is defined in terms of nodes and edges. (a). The undirected graph can only represent the dependencies between genes. Lines indicate the dependencies between A and B, and the possible directions are $A \rightarrow B$ and $B \rightarrow A$. (b). In the directed graph, two genes (A, B) are connected by an arrow, that is A is regulated by B.

relationships among genes, thus they do not reveal the causality between genes as shown in Figure 2.1.

For example, Figure 2.2 indicates a large-scale transcriptional gene regulatory network for yeast cell-cycle, which is composed of 3025 nodes and 6888 edges [39].

With respect to the computational approach for analysis of GRNs, over the last few decades, various studies for building GRNs have been conducted with the development of experimental techniques. (see details in Section 1.1). Of course, available information derived from genome expression profile is essential for reconstructing gene regulatory networks, at the same time, such "reverse-engineering" will be important to gain a better understanding of their behaviors. The analysis of GRNs will help us to gain new insights into biological complexity and flexibility behind not only gene expression but also biological systems. Consequently, the elucidation of gene regulatory networks is expected to provide a platform for medical treatments of genetic diseases, drug discovery and development. In this thesis, we will focus on how to reconstruct GRNs more reasonably and realistically from gene expression microarrays and present an unusual approach for achieving our goal.

2.1.1 Time-Varying Genetic Networks

Many biological processes are highly dynamic and exhibit temporal interactions between genes in response to intrinsic and extrinsic environmental cues. Time-varying gene regulatory networks are descriptions of dynamic behaviors of genetic networks underlying temporal biological processes, such as developmental process, immune re-

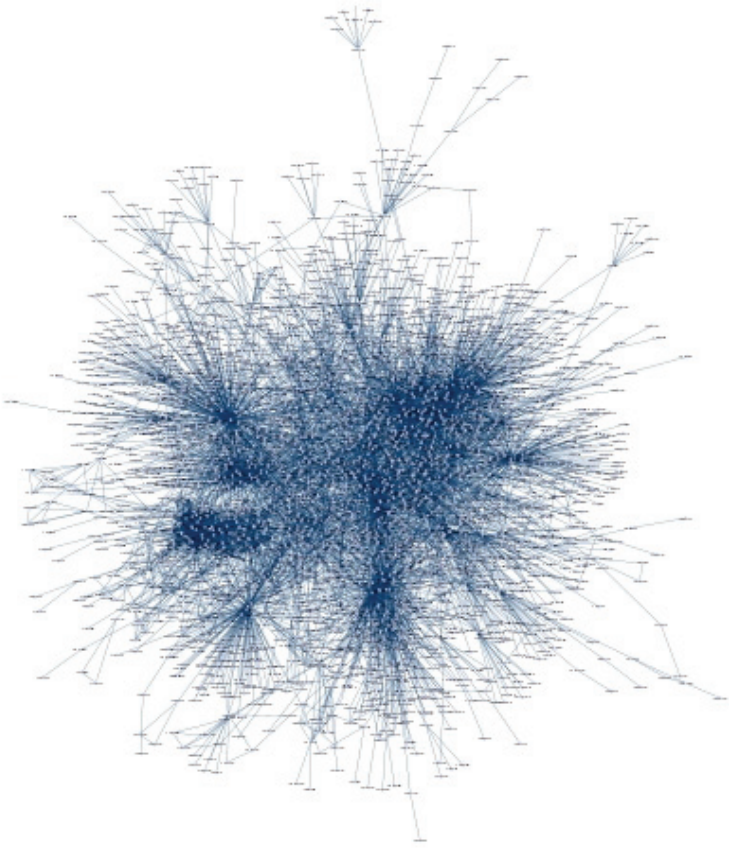


Figure 2.2: A large-scale gene regulatory network formed by cell-cycle transcription factors during the regulation of yeast cell-cycle progression.

sponse and disease progression [40]. For instance, the budding yeast, *Saccharomyces cerevisiae*, progresses through all distinct four stages during the cell-cycle; G1 (Gap1) phase, S (Synthesis) phase, G2 (Gap2) phase and M (Mitosis) phase shown in Figure 2.3.

The transitions between the phases are controlled by distinct gene regulatory networks induced by both environmental and endogenous stimuli [41]. During the phase transition, individual phase transition events occur independently in time-varying and exhibit temporally distinct gene expressions of phase-specific patterns associated with groups of functionally related genes. Through the transition events, the network is changing slowly over time [40]. In other words, the progressions of M and G1 phases are temporally controlled by different types of genetic interactions despite the existence of common sets of genes. This means that gene regulatory networks formed in the M phase is likely to be obviously different from that in G1 phase, even if a common set of genes are involved. As an another example, in the course of life-span development of fruit fly, *Drosophila melanogaster*, there exist four successive stages; embryonic, larval, pupal and adult and it is reported that these stages may also exhibit the same phenom-

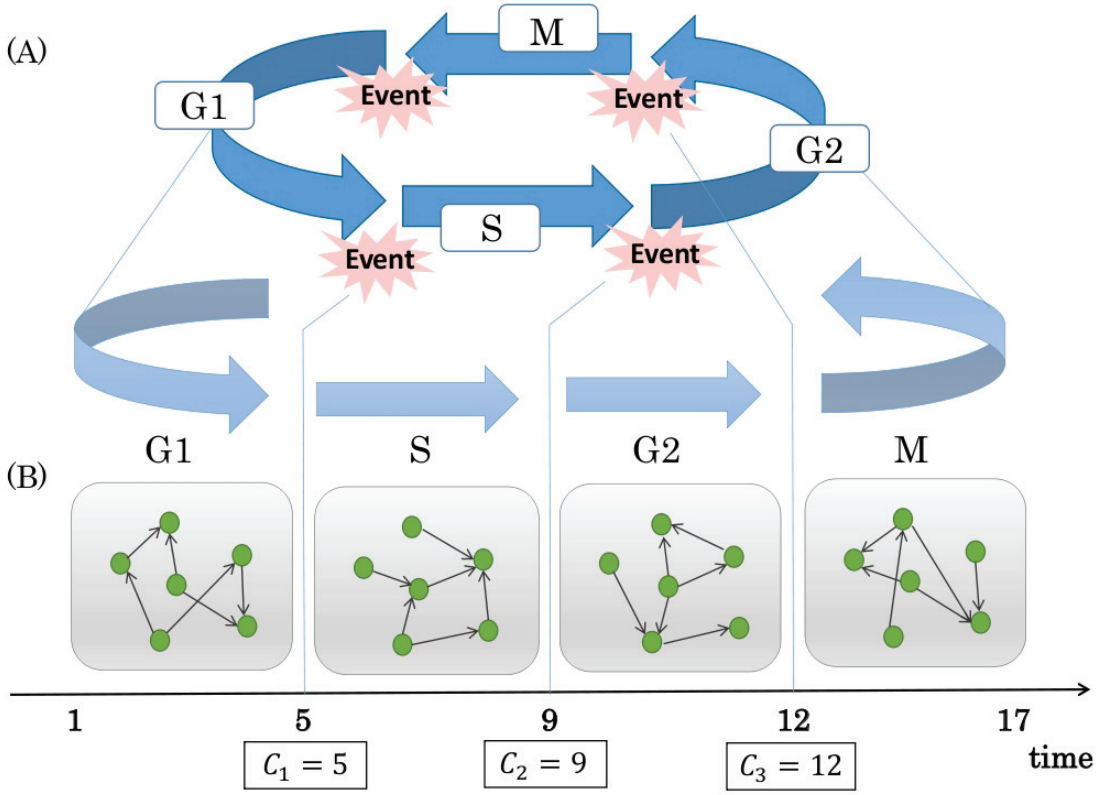


Figure 2.3: A schematic illustration of the time-varying network. The above diagram (A) indicates the course of the yeast cell-cycle. The diagram below (B) shows the alterations in gene regulatory networks consist of 6 nodes over time where C_i $(i = 1, 2, 3)$ represents the change point.

ena [40]. Thus, the alteration of cell function should be dependent on the architecture of gene regulatory networks that are directly or indirectly affected by internal and external stimuli over time. Therefore, genetic networks underlying temporal processes are likely to exhibit topological changes to facilitate the transition of regulatory functions varying in time.

The time-varying network also has an important feature, change points at which the network topology changes. As can be seen in Figure 2.3, there exist three change points over the course of the cell-cycle, and indeed network topologies change between the distinct phases. Change point detection problems have also been activity studied by modeling time-varying genetic networks (see Chapter 4 for detail) and analysis of time-varying networks may enable us to obtain an in-depth understanding of the dynamics and flexibility of gene regulatory systems.

2.1.2 DNA Microarray Technology

In the past few decades, scientists have been trying to solve the question: "What is the difference between healthy cells and normal cells in terms of genetics?" Previously, they could obtain information about gene expression patterns that involve only one or a few genes at a time. However, in the mid-1990s, DNA microarray technology has made a significant contribution to the field of biological studies. The development of DNA microarray has made it possible to simultaneously measure the concentration of mRNA for thousands of genes. Thus, the concentration of a particular mRNA reflects the expression levels of its corresponding gene [42]. The core principle behind microarrays is the hybridization that is the process where two complementary DNA strands taken from different sources bind to each other using base-pairing [43, 44]. For instance, a specific sequence in DNA "ATGACGT" will bind to the complementary sequence "TACTGCA" with a binding affinity.

A microarray consists of thousands of spots, each of which represents a different DNA and contains many copies of a particular DNA sequence which correspond to a specific gene. Currently, there are two types of DNA microarrays according to the probes spotted on the microarray; complementary DNA based microarray and oligonucleotide based microarray [45]. cDNA microarray contains thousands of the cDNA sequences representing the mRNAs that extracted from cell samples and they will hybridize to their complementary DNA strands (specific spots) on the microarray [46]. Consequently, a single microarray experiment can reveal mRNA expression levels of ten thousand of genes at once. Therefore, microarray analysis helps scientists to investigate the relationship in gene expression patterns of different cell or tissue types, such as healthy or cancer tissue, by comparing the expression profiles for every gene from a single experiment. It has been practically reported that information obtained from microarrays has been successfully used for disease diagnosis and gene discovery [44, 47].

Typically, microarray profiles can be divided into two types; time-series data and static data. A time-series is a temporal data, which consists of a set of observations measured at successive time instants spaced at uniform time intervals in the same sample [48, 49]. It provides the opportunities to understand and model the temporal dynamics of behaviors and relationships between genes underlying temporal biological processes. In the field of systems biology, the time-series data is known to be a powerful tool for reconstruction of genetic networks whose topologies change over time. Another type of data consists of a snapshot of gene expression at a single time point taken from different and independent samples [50]. This type of expression profile can usually be obtained from microarray experiments conducted on the disease progression and with genetic knockout organisms, and will be mostly static. Because, it is often the case that the cancer or other disease data cannot be obtained following the entire course of disease progression from the same patient. The knockout experiments are also carried

out to investigate the functions of specific genes by artificially removing them from the genome. Thus, static data can be expected to provide useful information on disease diagnosis through the analysis of gene regulatory networks under specific disease/normal conditions. Another difference between time-series and static data is that static data taken from samples are independently and identically distributed, whereas time-series data generally exhibits an autocorrelation between successive measurements [48]. Both types of microarray datasets are publicly available and have been widely used for the analysis of several genetic regulatory networks.

The aim of this thesis is to complete and infer regulatory interactions between genes related to specific biological processes in the form of a genetic network based on the known genetic network obtained from KEGG [51] or other databases and the corresponding gene expression data. So, we have to use the expression levels of a set of candidate genes already known to be involved in a particular biological process regardless of the type of experimental conditions (time-series or static). Based on our needs, microarray expression profiles are suitable observations.

2.2 Mathematical Optimization

This section describes the mathematical optimization. After a brief introduction to the optimization problem, we will focus on one topic in combinatorial optimization. Subsequently, we will explain the idea of dynamic programming paradigm for solving this optimization task. Finally, least-squares optimization will be considered, giving a detailed explanation of computational complexity analysis.

2.2.1 Introduction to Optimization Problem

We can routinely encounter optimization problems in many areas of real world, such as industry, science, mathematics, business and economics. Industrial engineers have widely developed approaches for product, material and process design, logistic and even strategic planning [52]. Mathematically, the ancient Greeks solved optimization problems in relation to geometry studies. For instance, Euclid proved that if a square and a rectangle have the same perimeters, the area of the square is larger than that of the rectangle and invented geometric theories. The discovery of calculus in the late 1600s allowed them to consider more complex problems, and just after the World War II, the theory and practice of optimization have been expanded with the development of operations research. Since the early 1960s, the complexity analysis of computational algorithms had begun to backup optimization uses, which could also make it possible to introduce the computer-based optimization. As computers became more efficient and powerful in the 1980s, optimization algorithms have been developed for the purpose of solving large-scale problems [53].

If there is a certain problem, the way to solve the given problem is firstly, to formulate a mathematical description for representing the situation called a mathematical model, which consists of a single objective function and a set of constraints. Secondly, to find the "best" or "optimal" solution that minimizes or maximizes the objective function from all feasible solutions which are the set of decision variables (see below) satisfying all constraints. Particularly, a mathematical optimization model consists of the following components [54].

Decision variable

The decision variable represents unknown quantities, for instance the number of items, time, profit etc., assigned mathematical symbols such as x_1, x_2. A solution is some assignment of values to the decision variables.

Objective function

The objective function is to express the goal driven optimization problem in mathematical form with decision variables. The objective may be minimizing costs, time, waste, distance, or maximizing profit, etc.

Constraint

The constraint reflects the limitation or requirement which a solution to the given problem must satisfy, and it is formulated as equations or inequalities with decision variables.

A feasible solution is some assignment of values to the decision variables that satisfies the constraints. More precisely, an optimization problem can be defined by the set of feasible solutions S and an objective function $f : S \to \mathbb{R}$. Therefore, the formal definition of optimization problems is as follows:

$$\begin{aligned} \min_{x} \quad & f(x), \\ \text{subject to} \quad & x \in S. \end{aligned} \tag{2.1}$$

The objective is to identify the best or optimal solution, $x \in S$ in minimizing (resp., maximizing) $f(x)$ subject to the constraint $x \in S$, that is, the solution $f : x \in S$ such that $f(x) \leq f(y)$ (resp., $(f(x) \geq f(y))$ for all $y \in S$.

Optimization problems can be classified into two groups according to the mathematical characteristics of decision variables and constraints. With regard to decision variables, whether the set of feasible solutions S is continuous or discrete is the basis for the classification of the continuous or the discrete problem. In continuous optimizations, the definition of the feasible solution set S can be represented by continuous or infinite number of variables. For instance, an optimization model with continuous variables where the objective function and constraints use only linear equations is known to

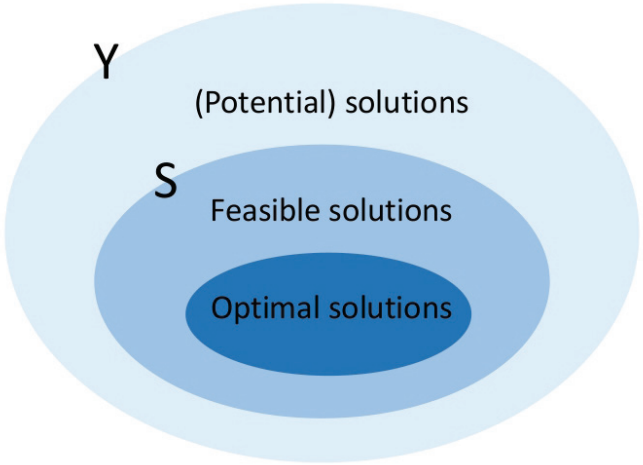


Figure 2.4: The inclusive relations among the solutions to the optimization problem. An optimal solution is a feasible solution that satisfies the objective function.

be *Linear Programming*. In contrast, a discrete optimization problem is also known as a combinatorial optimization problem (for details, refer to Section 2.2.2). Also for the nature of constraints, unlike the constrained optimization, the unconstrained optimization has no limitation imposed on decision variables. The most popular unconstrained optimization is data fitting problems (linear and nonlinear least-squares problems, regression models, etc.) [55]. In the following subsections, we will present examples for each constrained and unconstrained optimization problems.

2.2.2 Combinatorial Optimization

The combinatorial optimization means finding an optimal or near optimal solution among a finite number of feasible solutions that satisfy all the constraints. Interestingly, many real-life problems such as planning and scheduling can be modelled as combinatorial optimization problems [52, 56] and they are well-studied as *travelling salesman problems, knapsack problems, shortest spanning tree, network optimization problems*, etc. Our proposed problem network completion, is more likely to be similar to knapsack problems, so we will introduce the knapsack problem as an example of combinatorial optimization.

Example: We are given a knapsack that has a capacity (weight) W and a set of N items, where each item i ($i = 1, 2, \cdots, N$) has its own weight w_i and a profit p_i as presented in Table 2.1. The goal is to choose a set of items so as to maximize the total profit without exceeding the knapsack capacity W. Now consider $W = 30$.

Table 2.1: List of items in case of $N = 4$.

Item i	Weight w_i	Profit p_i
1	10	40
2	2	55
3	30	100
4	20	80

The chosen items must not exceed the knapsack capacity $W = 30$, thereby providing the feasible solutions, $\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 4\}, \{2, 4\}$. Therefore, we can easily determine that the best solution uses item 2 and 4, which gives the total profit 135 from all feasible solutions.

Formally, a combinatorial optimization is defined by a tuple $I = (Y, f, Z)$, where a finite set of potential solutions Y (see Figure 2.4) and an objective function $f : Y \rightarrow \mathbb{R}$ and the feasibility predicate Z to determine whether the solution is feasible (i.e., the item fits into the knapsack) or not. Note that x is a vector of the decision variables $\{x_1, x_2, \cdots, x_N\}$. The objective is to find a solution $x \in Y$ that maximizes (or minimizes) the objective function $f(x)$ [57]. Based on these definitions, the most general form of the knapsack problem can be described as

$$\begin{aligned} \max_{x} \quad & f(x) = \sum_{i=1}^{N} p_i x_i, \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i x_i \leq W, \\ & x_i \in \{0, 1\} \quad i = 1, 2, \cdots, N, \end{aligned} \tag{2.2}$$

where x_i can take only on the value either 0 or 1. This means that we can solve the 0/1 knapsack problem in exponential running time $O(2^N)$ with the size of the input by enumerating all possible solutions. Therefore, such a problem may seemingly appear to be trivial, because it is likely that an optimal solution will be found by enumerating a finite number of feasible solutions. However, if a given problem has too many possible solutions, this may result in a combinatorial explosion. This means that the time complexity required to solve the given problem continues to grow exponentially as the input size increases. Therefore, it is needed to develop algorithms that run in time which is a polynomial in the input size. Actually, combinatorial optimization may be divided into two types; "easy" and "hard" problems that stand for solving in polynomial time and in no polynomial time. Of course, it is always best to solve combinatorial optimization problems in polynomial time, but the fact also remains

that there exist several no polynomial time algorithms for such hard problems. The following subsection describes dynamic programming that is one of the most useful algorithmic paradigms for solving combinatorial optimization problems.

2.2.3 Dynamic Programming

Dynamic programming is a paradigm of algorithm design for solving optimization problems, and it was invented by Richard Bellman in the mid-1950s [58]. The "programming" is not meant to be writing code, but rather in the sense of planning or scheduling for military logistics, by filling in a table. Also for "dynamic" indicates that the table is filled over time [59]. Basically, the dynamic programming finds an optimal solution to the problem by combining optimal solutions to subproblems in a certain way. The key idea behind dynamic programming is to avoid unnecessary recursive function calls by storing (memorizing) the solutions of intermediate subproblems. It must be noted that dynamic programming can often be applied to most problems that exhibit two properties of optimal substructure and overlapping subproblems. The optimal substructure means that an optimal solution to an original problem always contains optimal solutions for smaller subproblems. In addition, if the problem exhibits overlapping subproblems, there is no need to recalculate the same subproblem which reappears more than once by storing its solution.

The procedure of dynamic programming consists of three main steps [60]; 1. Initialization; 2. Matrix fill; and 3. Traceback. Initialization is to create a matrix with a specific size and to set the first column and row of the matrix. The next step is filling up the matrix with the solutions by solving subproblems from upper left to lower right, and finally through the traceback, determine a best path giving the optimal solution. This procedure is done by looking at the elements representing optimal solutions to already considered subproblems which are located to the left, above and diagonal in the matrix. In the following, we try to solve the knapsack problem using dynamic programming.

Example: Suppose we have a knapsack whose capacity $W = 10$ and 4 items. Each item has its own weight $w[i]$ and profit $p[i]$ $(i = 1, \cdots, n)$ shown in the Table 2.2.

Table 2.2: List of n (i.e., $n = 4$) items.

Item i	Weight $w[i]$	Profit $p[i]$
1	3	7
2	2	4
3	2	8
4	6	9

Now, let us consider the knapsack problem defined by term (2.3),

$$\begin{aligned} \text{maximize} \quad & 7x_1 + 4x_2 + 8x_3 + 9x_4, \\ \text{subject to} \quad & 3x_1 + 2x_2 + 2x_3 + 6x_4 \leq 10, \\ & x_1, x_2, x_3, x_4 \in \{0, 1\}, \end{aligned} \tag{2.3}$$

where $x_i = 1$ if the item i is selected and $x_i = 0$ if not selected into the knapsack whose capacity is w ($1 \leq w \leq W$). For this problem, we create a 2-dimensional array D of size $(n + 1) \times (W + 1)$ (i.e., $n = 4$ and $W = 10$).

As the first step, we define $D[i, w]$ stores the profit of the most profitable subset of items by selecting from the first i items that fit into the knapsack of capacity w. If $w[i] \geq w$ that is, the weight of the i-th item $w[i]$ is greater than the least capacity w, it is impossible to add the i-th item into the knapsack for obtaining the total weight w. Thus, the profit of the subset by selecting from the first i items is the same as that from the first $i - 1$ items. Otherwise, if the i-th item should be added, the optimal subset must be made up of this item and the subset of the first $i - 1$ items that obtain the total profit $w - w[i]$ and $D[i-1, w-w[i]]$ is regarded as the least capacity of knapsack with $i-1$ items. Thus, the profit given such an optimal subset is $D[i-1, w-w[i]]+p[i]$. This selection can be defined by the following recurrence formula,

$$D[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0, \\ D[i-1, w] & \text{else if } w[i] \geq w, \\ \max\{D[i-1, w], D[i-1, w-w[i]] + p[i]\} & \text{otherwise.} \end{cases} \tag{2.4}$$

The goal of this problem is to determine $D[n, W]$, the maximum profit of a subset of the n items that fit into the knapsack with maximum capacity W and the optimal subset. According to Equation (2.4), each entry $D[i, w]$ can be computed in the i-th row and the w-th column as presented in Table 2.3.

Table 2.3: The dynamic programming table for solving the knapsack problem.

i \ w	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	7	7	7	7	7	7	7	7
2	0	0	4	7	7	11	11	11	11	11	11
3	0	0	8	8	12	15	15	19	19	19	19
4	0	0	8	8	12	15	15	19	19	19	21

We can obtain an optimal solution whose maximum profit is 21 by selecting item 2, 3 and 4. Basically, the 0/1 knapsack problem can be solved in exponential time by enumerating all possible subsets over the n items, as previously mentioned. Using dynamic programming, however, it can reduce the time complexity to $O(nW)$, in which the table D has $(n+1) \times (W+1)$ entries and each of them can be computed in $O(1)$ time. Thus, the dynamic programming algorithm for the 0/1 knapsack problem can relatively decrease the time complexity. Actually, $O(nW)$ does depend polynomially on W, the issue is that $O(nW)$ does not depend polynomially on the size of the input of the problem. Because each weight can be represented using $O(\log W)$ bits [61], the time complexity is $O(nW) = O(n2^S)$ by letting $S = \log W$. Similarly, the longest common subsequence problem can be solved efficiently using dynamic programming. This problem is to find the maximum length of a common subsequence of two strings and is used in biological applications like comparing the two different strands of DNA [62].

In this thesis, we will present network completion as an optimization problem and this problem exhibits the optimal substructure (see Chapter 3 for the exact definition). The reason comes from that it can be decomposed into many subproblems that are replaced by smaller problems of completing networks containing a smaller number of genes. Moreover, this problem also exhibits the overlapping subproblems because the same combinations of incoming edges exist when computing the minimum sum-of-squared errors. In network completion, we will regard the total number of modified (added and deleted) edges and the sum-of-squared errors as the total weight and the total profit defined in the knapsack problem. In addition, since the total number of modified edges is specified in network completion, it can be considered as the optimization problem of adding and deleting edges for a given network such that the sum-of-squared error is minimized without exceeding the total number of modified edges. The correctness of dynamic programming and the detailed formulation are presented in later chapters.

2.2.4 Least-Squares Optimization

Least-squares problems may be formulated as an unconstrained continuous optimization, where the objective function is defined as the sum-of-squared vertical distances between the data points and the fitted line. Least-squares techniques have been commonly used for the statistical analysis of experimental or observational data. Least-squares problems can be classified into two types; linear (ordinary) or nonlinear least-squares according to whether the property of the objective function is linear or nonlinear. Now, we will only consider the linear least-squares problem. Given a set of observations and a model function, the objective of this problem is to determine unknown estimators of model function so as to minimize the objective function. In the following, we will also describe how to solve this problem, giving a simple example.

Example: We will consider the simplest linear model written in slope-intercept form;

$$Y = \beta_0 + \beta_1 X + \xi, \tag{2.5}$$

where X is the *independent* (or *explanatory*) variable and Y is the *dependent* (or *response*) variable. In general, suppose that only dependent variables Y are subject to measurement errors ξ and can be written as a linear regression function of the independent variables X. Thus, β_1 and β_0 are the slope and the intercept as shown in Figure 2.5.

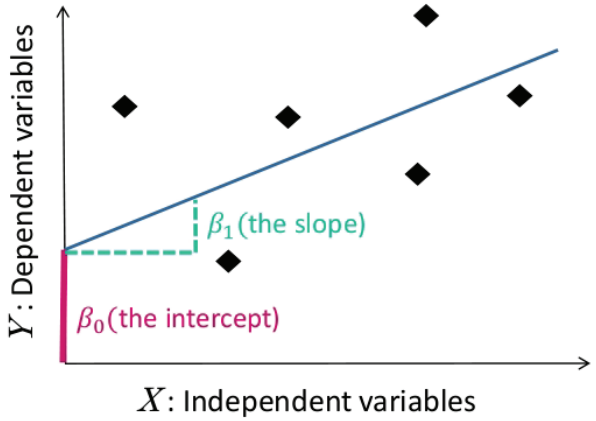


Figure 2.5: The simplest least-squares problem is to determine the straight line which has slope β_1 and y-intercept β_0.

Suppose we have N pairs of observations, (x_i, y_i) $(i = 1, 2, \cdots, N)$, the regression line is given by the following formula:

$$y_i = \beta_0 + \beta_1 x_i + \xi_i. \tag{2.6}$$

In this simple case, set $\xi_i = 0$, the goal of least-squares problem is to determine the least-squares estimators for β_0 and β_1 that minimize the sum-of-squared residuals from the observations, which is defined by Equation (2.7).

$$S(\beta_0, \beta_1) = \min \sum_{i=1}^{N} (y_i - (\beta_0 + \beta_1 x_i))^2. \tag{2.7}$$

For the analysis by linear regression models, it will be convenient to express them in matrix notation as follows.

$$\hat{u} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_N \end{pmatrix}, y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, X = \begin{pmatrix} 1 & x_{12} \\ 1 & x_{22} \\ \vdots & \vdots \\ 1 & x_{N2} \end{pmatrix}, \hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}. \tag{2.8}$$

Let $\hat{u_i}$ $(i = 1, \cdots, N)$ be the vertical distances between the observed and the predicted values. Note that $\hat{u}$ and y are $N \times 1$ vectors and the least-squares estimator $\hat{\beta}$ is denoted by $P \times 1$ vector $(P = 2)$. It is also noted that a design matrix X has an $N \times P$ dimension where the first independent variable x_{i1} is equal to 1 for every i $(i = 1, \cdots, N)$ because β_0 is a constant. Thus, we can obtain the following equation by rewriting Equation (2.6) in matrix notation.

$$\hat{u} \quad = \quad y - X\hat{\beta}. \tag{2.9}$$

In addition, the squared residual is also given by

$$\|\hat{u}\|^2 \quad = \quad \left(y - X\hat{\beta}\right)' \left(y - X\hat{\beta}\right). \tag{2.10}$$

Note that for a matrix A, its transpose is denoted by A'. Therefore, the least-squares problem can be rewritten as Equation (2.11),

$$S\left(\hat{\beta}\right) = \min_{\hat{\beta}} \left\|y - X\hat{\beta}\right\|^2. \tag{2.11}$$

The least-squares estimator $\hat{\beta}$ can be obtained by minimizing $S\left(\hat{\beta}\right)$, we then set the derivative of $S\left(\hat{\beta}\right)$ in terms of $\hat{\beta}$ equal to zero;

$$\frac{\partial S\left(\hat{\beta}\right)}{\partial \hat{\beta}} = -2X'\left(y - X\hat{\beta}\right) = 0. \tag{2.12}$$

Solving this for $\hat{\beta}$, we obtain

$$\hat{\beta} = \left(X'X\right)^{-1}\left(X'y\right). \tag{2.13}$$

Note that since there exists an inverse of $X'X$, the matrix dimension of X should be $N \times P$, and this means that the number of parameters P must be less than or equal to the amount of training data N (see [63] for more details).

In this thesis, we consider the problem of network completion where the edges in a given network are modified so as to match the observed data. To achieve our purpose, it requires introduction of more effective function to minimize the difference between observations and predictions. If we define an objective function as the sum-of-squared residuals when adding and deleting edges, network completion can be reduced to an optimization problem of minimizing the objective function. Three problems need their own objective functions and the detailed description of each problem will be provided in later chapters.

Following this, we will discuss the computational complexity required for the algorithm for finding the least-squares estimator $\hat{\beta}$. The execution time taken by running a program depends on the number of floating point operations (flops) [64, 65]. It is mainly estimated by counting the four elementary operations. We will count here only the number of multiplication operations.

Now, given the linear least-squares estimator defined in Equation (2.13), the computation of this system is divided into 4 operations, which involved 3 matrix multiplications and 1 matrix inversion.

In order to estimate an overall time complexity, we need to compute their own complexities of 4 operations considering the following points. The multiplication of two $n \times n$ matrices takes $O(n^3)$ time. In matrix-vector multiplication, assuming that the matrix is $n \times m$, then the vector must have a dimension m, thereby taking the time complexity of $O(nm)$.

If there are N training samples and P parameters, a matrix X should obviously have size $N \times P$. Since the least-squares fitting requires 4 operations as described above, based on these assumptions, their own complexities can be estimated in the following manner [65, 66].

1. Multiplying X' by X takes $O(NP^2)$ time.

2. The resulting matrix from Step 1 has dimension $P \times P$, thereby providing the complexity of inverting $X'X$ via LU decomposition scales as $O(P^3)$.

3. Multiplication of the matrix X' whose size is $P \times N$ and a vector y has a length N can be performed in $O(NP)$.

4. The product of two matrices $(X'X)^{-1}(X'b)$ resulting from Step 2 and 3 has a complexity $O(P^2)$.

Note that $O(NP^2)$ asymptotically dominates $O(NP)$, thus we can ignore the $O(NP)$ part and $O(P^3)$ also dominates $O(P^2)$. Consequently, the overall time complexity of this algorithm can be estimated as $O(NP^2 + P^3)$. The execution time is very slow when the P is large, however, the use of a large P gives a better approximation.

Chapter 3

Network Completion Using Dynamic Programming and Least-Squares Fitting

3.1 Background

Analysis of biological networks is one of the central research topics in computational systems biology. A number of reverse-engineering methods have been extensively developed to build genetic networks from microarray time-series. Most of these methods include general network models such as Boolean networks [5, 67], Bayesian networks [7, 9], time-delayed Bayesian networks [68], dynamic Bayesian networks [13, 14, 15], graphical Gaussian models [69, 70, 71], differential equations [18, 19], mutual information [27, 28], and linear classification [72]. However, these approaches may appear to be unrealistic to reconstruct genetic networks based only on information derived from gene microarray profiles and there is no gold standard or no established method for genetic network inference. At present, this research topic still remains challenging. One possible causes of this difficulty may be considered that there do not currently exist enough high-quality data to capture regulatory relationships in gene expression. This means that reverse-engineering of complex genetic regulations from only sparse and noisy data may lead to intrinsic difficulty, thereby inducing poor reconstruction accuracy. Therefore, a more reasonable and realistic approach is needed to infer gene regulatory networks.

Recently a new approach called network completion [37] was proposed by following Occam's razor, which is a well-known principle in scientific discovery. Network completion is, given an initial network and an observed dataset, to make the minimum amount of modifications to an initial network such that the resulting network is (most) consistent with the observed expression data. It focused on the inference of signaling pathway networks, which assumed that activity levels or quantities of one or a few kinds of pro-

teins can only be observed. Furthermore, a Boolean network [73] was used for modeling of signaling networks, because microarray might usually contain high level of noise and they were also concerned with theoretical analysis of computational complexity. They proved that network completion is computationally intractable (NP-hard) even for tree structured networks. In order to cope with this computational difficulty, an integer linear programming-based method was developed for completion of signaling pathways [74]. However, it could not handle addition of edges because of its high computational complexity.

In this chapter, we present a novel method, named as DPLSQ(Dynamic Programming and Least-SQuares), for network completion of gene regulatory networks using microarray time-series data. Unlike the previous studies [37, 74], we use a differential equation based model and assume that expression levels are fully observable. DPLSQ is based on the combination of least-squares fitting and dynamic programming techniques, where least-squares fitting is used to estimate parameters in differential equations and dynamic programming is used to find the minimum value of sum-of-squared errors by integrating partial fitting results (partial least-squares) on individual genes under the constraint that the total number of added and deleted edges must be equal to specified ones. One of the most important characteristics of DPLSQ is that it can provide an optimal solution (i.e., minimum squared-sum) in polynomial time subject to the constraint that the maximum indegree (i.e., the maximum number of input genes) is bounded by a constant. Although DPLSQ does not automatically identify the minimum modification, it can be identified by examining varying numbers of added/deleted edges, where the total number of such combinations is polynomially bounded. In addition, DPLSQ can also be used as a tool for genetic network inference, given a null network (i.e., a network having no edges) as an initial network.

We perform the computational experiments to validate the effectiveness of DPLSQ and also conduct the performance evaluation by comparing existing tools for network inference using artificially generated data. Furthermore, to evaluate our performance on real microarray data, we conduct network inference analysis on microarray profiles measured during the cell-cycle in *Saccharomyces cerevisiae*.

3.2 Method

In this section, we will begin by describing the exact definition of the network completion problem and a novel computational method DPLSQ, for solving this problem.

3.2.1 Preliminaries

The goal of network completion problem is to modify a given network with the minimum number of modifications such that the resulting network is most consistent with

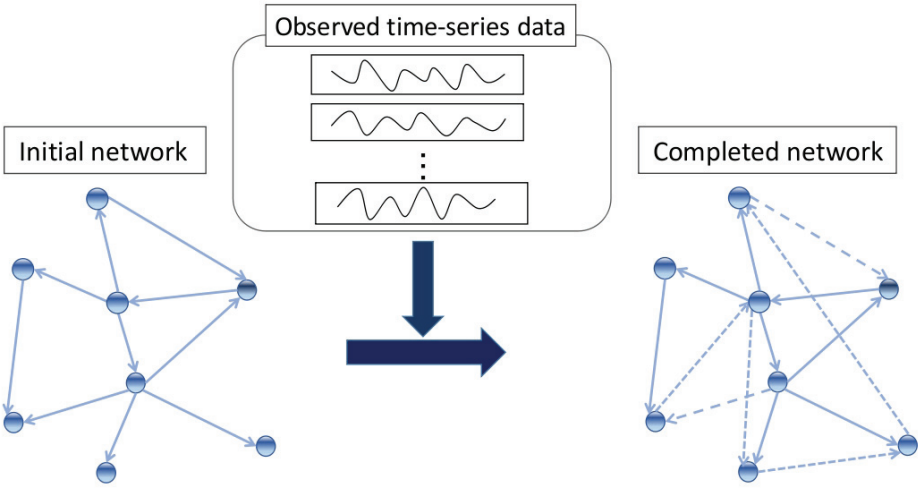


Figure 3.1: Schematic view of the network completion by additions and deletions of edges. Dashed edges and dotted edges denote deleted edges and added edges, respectively.

the observed data. For this problem, we consider additions and deletions of edges as modification operations (see Figure 3.1) and we regard the time-series microarray profile as an observed data. Note that we assume that the total number of added/deleted edges are specified. If we begin with a null network, it corresponds to network inference. Therefore, network completion includes network inference although it may not necessarily work better than existing methods if applied to network inference.

In the following, $G(V, E)$ denotes a given network where V and E are the sets of nodes and directed edges, where nodes reflect genes and edges represent causal regulatory effects between two genes. Self-loops are not allowed in E although it is possible to modify the method so that self-loops are allowed. In this thesis, let n denote the number of genes (i.e., the number of nodes) and let $V = \{v_1, \ldots, v_n\}$. For each node v_i, $e^-(v_i)$ and $deg^-(v_i)$ denote respectively the set of incoming edges to v_i and the number of incoming edges to v_i as defined below;

$$\begin{aligned} e^-(v_i) &= \{v_j | (v_j, v_i) \in E\}, \\ deg^-(v_i) &= |e^-(v_i)|. \end{aligned} \tag{3.1}$$

DPLSQ consists of two parts: (i) parameter estimation, and (ii) network structure inference. We employ least-squares fitting for the former part and dynamic programming for the latter part. Furthermore, there are three cases on the latter parts: (a) completion by addition of edges, (b) completion by deletion of edges, and (c) completion by addition and deletion of edges. Although the last case includes the first and second cases, we explain all of these for the sake of simplicity of explanation.

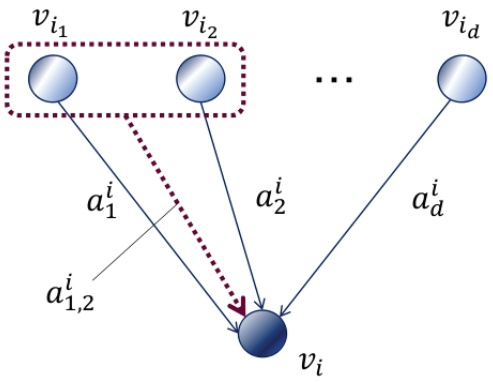


Figure 3.2: Dynamics model for a node. The expression level of v_i is determined by the correlation between other genes (i.e., $v_{i_1}, \cdots, v_{i_d}$). $a^i_{1,2}$ is a coefficient corresponding to cooperative regulation by v_{i_1} and v_{i_2}.

3.2.2 Model of Differential Equation and Estimation of Parameters

We assume that the differential equation model for the dynamics of each node v_i can be written as Equation (3.2),

$$\frac{dx_i}{dt} = a^i_0 + \sum_{j=1}^{d} a^i_j x_{i_j} + \sum_{1 \le j < h \le d} a^i_{j,h} x_{i_j} x_{i_h} + b^i \omega, \tag{3.2}$$

where $v_{i_1}, \ldots, v_{i_d}$ are incoming nodes to v_i, x_i corresponds to the expression value of the i-th gene, and ω denotes a random noise. The second and third terms of the right hand side of the equation represent linear and non-linear effects to node v_i, respectively (see Figure 3.2), where positive a^i_j or $a^i_{j,h}$ reflects to an activation effect and negative a^i_j or $a^i_{j,h}$ reflects to an inhibition effect.
In practice, we replace the derivative by the difference and ignore the noise term as below:

$$x_i(t + \Delta t) = x_i(t) + \Delta t \left(a^i_0 + \sum_{j=1}^{d} a^i_j x_{i_j}(t) + \sum_{1 \le j < h \le d} a^i_{j,h} x_{i_j}(t) x_{i_h}(t) \right). \tag{3.3}$$

We assume time-series data $y_i(t)$s, which correspond to $x_i(t)$s are given for $t = 0, 1, \ldots, m$. Then, we can use the standard least-squares fitting to estimate parameters a^i_js and $a^i_{j,h}$s.

By introducing least-squares fitting technique, network completion can be reduced to the optimization problem to minimize the following objective function:

$$S^i_{\{v_{i_1}, v_{i_2}, ..., v_{i_d}\}} = \min_{a^i_0, a^i_j, a^i_{j,h}} \sum_{t=0}^{m} \left| y_i(t+\Delta t) - \left[y_i(t) + \Delta t \left(a^i_0 + \sum_{j=1}^{d} a^i_j y_{i_j}(t) + \sum_{1 \le j < h \le d} a^i_{j,h} y_{i_j}(t) y_{j_h}(t) \right) \right] \right|^2 . \tag{3.4}$$

3.2.3 Completion by Addition of Edges

In this subsection, we consider the problem of adding h edges in total so that the sum-of-squared errors is minimized.

Let $\sigma^+_{h_j, j}$ denote the minimum sum-of-squared error when adding h_j edges to the j-th node, which is formally defined by

$$\sigma^+_{h_j, j} = \min_{j_1, j_2, ..., j_{h_j}} \left\{ S^j_{e^-(v_j) \cup \{v_{j_1}, v_{j_2}, ..., v_{j_{h_j}}\}} \right\}, \tag{3.5}$$

where each v_{j_l} must be selected from $V - v_j - e^-(v_j)$. In order to avoid combinatorial explosion, we constrain the maximum h_j to be a small constant H and let $\sigma^+_{h_j, j} = +\infty$ for $h_j > H$ or $h_j + deg^-(v_j) \ge n$. Then, this problem is stated as term (3.6),

$$\min_{h_1 + h_2 + \cdots + h_n = h} \sum_{j=1}^{n} \sigma^+_{h_j, j}. \tag{3.6}$$

Here, we define $D^+[h, i]$ by

$$D^+[h, i] = \min_{h_1 + h_2 + \cdots + h_i = h} \sum_{j=1}^{i} \sigma^+_{h_j, j}. \tag{3.7}$$

Then, $D[h, n]$ is the objective value (i.e., the minimum sum-of-squared error when adding h edges in total).

The elements of $D[h, j]$ can be computed by the following dynamic programming algorithm:

$$\begin{aligned} D^+[h, 1] &= \sigma^+_{h,\,1}, \\ D^+[h, j+1] &= \min_{h' + h'' = h} \{ D^+[h', j] + \sigma^+_{h'',\, j+1} \}. \end{aligned} \tag{3.8}$$

It is to be noted that $D^+[h, n]$ is determined uniquely regardless of the order of

nodes in the network. The correctness of this dynamic programming algorithm can be formally proven by

$$\begin{aligned}\min_{h_1+h_2+\cdots+h_n=h}\sum_{j=1}^{n}\sigma^{+}_{h_j,j} &= \min_{h'+h''=h}\left\{\min_{h_1+h_2+\cdots+h_{n-1}=h'}\sum_{j=1}^{n-1}\sigma^{+}_{h_j,j}+\sigma^{+}_{h'',n}\right\},\\ &= \min_{h'+h''=h} D^{+}[h',n-1]+\sigma^{+}_{h'',n}.\end{aligned} \tag{3.9}$$

3.2.4 Completion by Deletion of Edges

We try to address the completion problem of deleting w edges in total so as to minimize the sum-of-squared errors.

Let $\sigma^{-}_{w_j,j}$ denote the minimum sum-of-squared error when deleting w_j edges from the set $e^{-}(v_j)$ of incoming edges to v_j as follows;

$$\sigma^{-}_{w_j,j} = \min_{j'_1,j'_2,\ldots,j'_{w_j}}\left\{S^{j}_{e^{-}(v_j)-\{v_{j'_1},v_{j'_2},\ldots,v_{j'_{w_j}}\}}\right\}. \tag{3.10}$$

As described in Section 3.2.3, we need to impose constraint on the maximum w_j to be a small constant W and let $\sigma^{-}_{w_j,j}=+\infty$ if $w_j > W$ or $deg^{-}(v_j)-w_j<0$. Then, the problem can be stated as term (3.11);

$$\min_{w_1+w_2+\cdots+w_n=w}\sum_{j=1}^{n}\sigma^{-}_{w_j,j}. \tag{3.11}$$

Here, we also define $D^{-}[w,i]$ by

$$D^{-}[w,i] = \min_{w_1+w_2+\cdots+w_i=w}\sum_{j=1}^{i}\sigma^{-}_{w_j,j}. \tag{3.12}$$

Then, we can solve network completion by deletion of edges using the following dynamic programming algorithm:

$$\begin{aligned}D^{-}[w,1] &= \sigma^{-}_{w,1},\\ D^{-}[w,j+1] &= \min_{w'+w''=w}\{D^{-}[w',j]+\sigma^{-}_{w'',j+1}\}.\end{aligned} \tag{3.13}$$

3.2.5 Completion by Addition and Deletion of Edges

We can extend the above two methods to network completion by addition and deletion of edges.

Let $\sigma_{h_j,w_j,j}$ denote the minimum sum-of-squared error when adding h_j edges to $e^{-}(v_j)$ and deleting w_j edges from $e^{-}(v_j)$ as below, where added and deleted edges

must be disjoint.

$$\sigma_{h_j,w_j,j} = \min_{\substack{j_1,j_2,\ldots,j_{h_j} \\ j'_1,j'_2,\ldots,j'_{w_j}}} \left\{ S^j_{e^-(v_j)-\{v_{j'_1},v_{j'_2},\ldots,v_{j'_{w_j}}\}\cup\{v_{j_1},v_{j_2},\ldots,v_{j_{h_j}}\}} \right\}. \tag{3.14}$$

Here, we also impose constraints on the maximum h_j and w_j to be small constants H and W. We let $\sigma_{h_j,w_j,j} = +\infty$ if $h_j > H$, $w_j > W$, $h_j - w_j + deg^-(v_j) \geq n$, or $w_j - h_j + deg^-(v_j) < 0$ holds. Then, the problem is stated as term (3.15);

$$\min_{\substack{h_1+h_2+\cdots+h_n=h \\ w_1+w_2+\cdots+w_n=w}} \sum_{j=1}^{n} \sigma_{h_j,w_j,j}. \tag{3.15}$$

We define $D[h,w,i]$ by

$$D[h,w,i] = \min_{\substack{h_1+h_2+\cdots+h_i=h \\ w_1+w_2+\cdots+w_i=w}} \sum_{j=1}^{i} \sigma_{h_j,w_j,j}. \tag{3.16}$$

Then, we can also solve network completion by addition and deletion of edges using the following dynamic programming algorithm:

$$\begin{aligned} D[h,w,1] &= \sigma_{h,w,1}, \\ D[h,w,j+1] &= \min_{\substack{h'+h''=h \\ w'+w''=w}} \{D[h',w',j] + \sigma_{h'',w'',j+1}\}. \end{aligned} \tag{3.17}$$

3.2.6 Time Complexity Analysis

In this subsection, we provide the time complexity analysis of DPLSQ. Since completion by addition of edges and completion by deletion of edges are special cases of completion by addition and deletion of edges, here we discuss only cases of completion by addition and deletion of edges.

Before discussing the time complexity taken for each step, we need to check the time complexity required for least-squares fitting. Least-squares fitting is generally known to have time complexity $O(mp^2 + p^3)$, assuming that m is the sample size and p is the number of predictors as indicated in 2.2.4. Since our model has $O(n^2)$ parameters per node, the time complexity for least-squares can be estimated as $O(mn^4 + n^6)$ time. It must be noted, however, if we can assume that the maximum indegree in a given network is bounded by a constant, the number of parameters is bounded by a constant, where we have already assumed that H and W are constants. In this case, the time complexity for least-squares fitting can be estimated as $O(m)$.

First, we analyze the time complexity required for computing $\sigma_{h_j,w_j,j}$. In this computation, we need to examine combinations of additions of h_j edges and deletions of w_j

edges. Since h_j and w_j are respectively bounded by constants H and W, the number of combinations is $O(n^{H+W})$. Therefore, the computation time required per $\sigma_{h_j,w_j,j}$ is $O(n^{H+W}(mn^4+n^6))$ including the time for least-squares fitting. Since we need to compute $\sigma_{h_j,w_j,j}$ for $H \times W \times n$ combinations, the total time required for computation of $\sigma_{h_j,w_j,j}$s is $O(n^{H+W+1}(mn^4+n^6))$.

The next step is the analysis of time complexity taken by computing $D[h,w,i]$s. Note that the size of table $D[h,w,i]$ is $O(n^3)$, where we are assuming h and w are $O(n)$. In order to compute the minimum value for each entry in the dynamic programming procedure, we need to examine $(H+1)(W+1)$ combinations, which is $O(1)$. Therefore, the computation time required for computing $D[h,w,i]$s is $O(n^3)$. Since this value is clearly smaller than the one for $\sigma_{h_j,w_j,j}$s, the total time complexity is,

$$O(n^{H+W+1} \cdot (mn^4+n^6)). \tag{3.18}$$

Although this value is too high, it can be significantly reduced if we can assume that the maximum degree of an input network is bounded by a constant. In this case, least-squares fitting can be done in $O(m)$ time per execution as discussed above. Furthermore, the number of combinations of deleting at most w_j edges is bounded by a constant. Therefore, the time complexity required for computing $\sigma_{h_j,w_j,j}$s is reduced to $O(mn^{H+1})$. Since the time complexity for computing $D[h,w,i]$s remains $O(n^3)$, the total time complexity is

$$O(mn^{H+1}+n^3). \tag{3.19}$$

This number may be allowable in practice if $H \leq 2$ and n is not too large (e.g., $n \leq 100$).

3.3 Results and Discussion

3.3.1 Availability

We validated the performance of proposed method using both synthetic data and microarray expression data. All experiments on DPLSQ were performed on a PC with Intel Core i7-2630QM CPU (2.00 GHz) with 8 GB RAM running under the Cygwin on Windows 7. We employed the liblsq library (http://www2.nict.go.jp/aeri/sts/stmg/K5/VSSP/install_lsq.html) for a least-squares fitting method.

3.3.2 Completion Using Synthetic Data

In order to evaluate the potential effectiveness of DPLSQ, we began with network completion using synthetic data. To our knowledge, there is no available tool that performs the same task. Although some of existing inference tools employ incremental

modifications of networks, the total number of added/deleted edges cannot be specified. Therefore, we did not compare DPLSQ with regard to network completion with other methods (but we compared it with existing tools for network inference).

We employed structure of the real biological network named WNT5A (see Figure 3.3) [75]. For each node v_i with d input nodes, we considered the following model:

$$x_i(t+\Delta t) = x_i(t) + \Delta t \left(a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j} + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j}(t) x_{i_h}(t) + b_i \omega \right), \tag{3.20}$$

where a_j^is and $a_{j,h}^i$s are constants selected uniformly at random from $[-1, 1]$ and $[-0.5, 0.5]$, respectively. The reason why the domain of $a_{j,h}^i$s is smaller than that for a_j^is is that non-linear terms are not considered as strong as linear terms. It should also be noted that $b_i\omega$ is a stochastic term, where b_i is a constant (we used $b_i = 0.2$ in all computational experiments) and ω is a random noise taken uniformly at random from $[-1, 1]$.

For artificial generation of observed data $y_i(t)$, we used

$$y_i(t) = x_i(t) + o^i \epsilon, \tag{3.21}$$

where o^i is a constant denoting the level of observation errors and ϵ is a random noise taken uniformly at random from $[1, -1]$. Since use of time-series data beginning from only one set of initial values easily resulted in rank deficiency, we generated time-series beginning from 20 sets of initial values taken uniformly at random from $[1, -1]$, where the number of time points for each set was equal to 10 and $\Delta t = 0.2$ was used as the period between the successive two time points. Therefore, 20 sets of time-series data, each of which consisted of 10 time points, were used per trial (200 time points were used in total per trial). Noted that here, in our preliminary experiments, use of too small Δt resulted in too small changes of expression values whereas use of large Δt resulted in divergence of time series. Therefore, after some trials, $\Delta t = 0.2$ was selected.

Under the above model, we examined several o^is as shown in Table 3.1. In order to examine network completion, an original network, WNT5A was modified by randomly adding h edges and deleting w edges and the resulting network was given as an initial network.

We evaluated the performance of DPLSQ in terms of the accuracy of the modified edges and the success rate. The accuracy is defined here by

$$\frac{h + w + |E_{orig} \cap E_{cmpl}| - |E_{orig}|}{h + w}, \tag{3.22}$$

where E_{orig} and E_{cmpl} are the sets of edges in the original network and the completed network, respectively. This value takes 1 if all deleted and added edges are correct

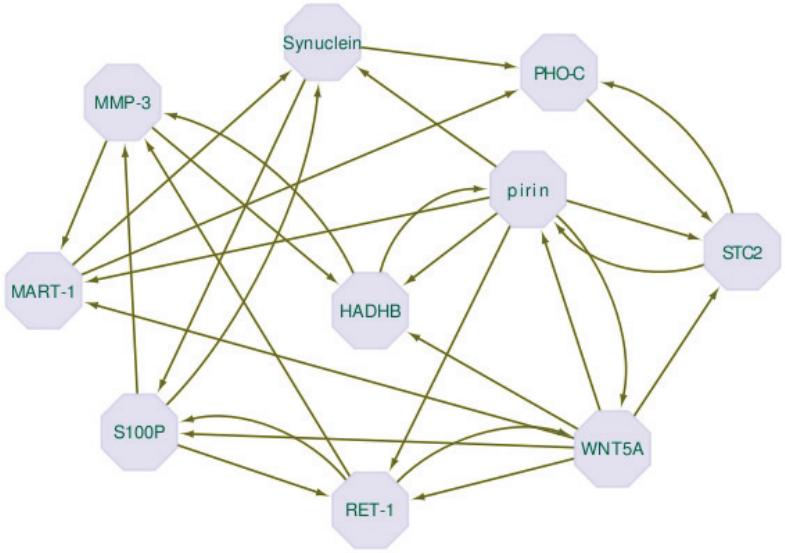


Figure 3.3: Structure of WNT5A network [75].

and 0 if none of deleted and added edges is correct. For each (h, w), we took the average accuracy over a combination of 10 parameters (a_j^is and $a_{j,h}^i$s) and 10 random modifications (i.e., addition of h edges and deletion of w edges to construct an initial network). The success rate is defined as the frequency of the trials (among 10×10 trials) in which the original network was fully correct obtained by network completion. The result is shown in Table 3.1. As can be seen from this table, DPLSQ works well if the observation error level is small. With regard to the accuracy, while these are high in the case of $w = 0$, no clear trend can be discerned for a relationship between h, w values and the accuracies. It is reasonable because we evaluated the result in terms of the accuracy per added/deleted edge. On the other hand, it is observed that the success rate decreases considerably as h and w increases or the observation error level increases. This dependence on h and w is reasonable because the probability of having at least one wrong edge increases as the number of edges to be added/deleted increases.

As for the computation time, CPU time for each trial was within a few seconds, where we used the default values of $H = W = 3$. Although these default values were larger than h, w here, it did not cause any effects on the accuracy or the success rate. How to choose H and W is not a trivial problem. As discussed in Section 3.2.6, we cannot choose large H or W because of the time complexity issue. Therefore, it might be better in practice to examine several combinations of small values H and W and select the best result although how to determine the best result is left as another issue.

Table 3.1: Results on completion of WNT5A network, where the average accuracy is shown for each case.

No. added edges	No. deleted edges		Observation error level			
			0.1	0.3	0.5	0.7
$h = 1$	$w = 0$	Accuracy	0.990	0.910	0.730	0.410
		Success rate	0.99	0.91	0.73	0.41
$h = 2$	$w = 0$	Accuracy	1.000	0.955	0.670	0.395
		Success rate	1.00	0.91	0.42	0.17
$h = 0$	$w = 1$	Accuracy	0.990	0.850	0.470	0.240
		Success rate	0.99	0.85	0.47	0.24
$h = 1$	$w = 1$	Accuracy	0.995	0.845	0.405	0.210
		Success rate	0.99	0.71	0.11	0.02
$h = 2$	$w = 1$	Accuracy	0.983	0.843	0.470	0.190
		Success rate	0.95	0.58	0.11	0.00
$h = 0$	$w = 2$	Accuracy	1.000	0.795	0.440	0.215
		Success rate	1.00	0.67	0.18	0.01
$h = 1$	$w = 2$	Accuracy	0.996	0.833	0.453	0.223
		Success rate	0.99	0.53	0.05	0.01
$h = 2$	$w = 2$	Accuracy	1.000	0.862	0.517	0.285
		Success rate	1.00	0.56	0.03	0.01

Table 3.2: Results on inference of WNT5A network by DPLSQ.

		Observation error level			
		0.1	0.3	0.5	0.7
$n = 10$	Accuracy	1.000	0.966	0.803	0.620
	CPU time (sec.)	0.685	0.682	0.682	0.685
$n = 30$	Accuracy	0.995	0.914	0.663	0.443
	CPU time (sec.)	66.2	66.2	66.1	65.9
$n = 50$	Accuracy	0.999	0.913	0.613	0.392
	CPU time (sec.)	534.0	534.2	533.6	533.5

3.3.3 Inference Using Synthetic Data

We also examined DPLSQ for network inference, using synthetic time-series data. In this experiment, we used the same network and dynamics model as above but we let $E = \emptyset$ in the initial network. In order to apply DPLSQ to network inference, we let $H = 3$, $W = 0$, and $h = 30$. It is to be noted that $deg^-(v_i) = 3$ holds for all nodes v_i in the WNT5A network. Furthermore, in order to examine how CPU time changes as the size of the network grows, we made networks with 30 genes and 90 edges (resp., with 50 genes and 150 edges) by making 3 copies (resp., 5 copies) of the original networks.

Since the number of added edges always should be equal to the total number of edges in the original network, h, we evaluated the results by the average accuracy defined as the ratio of the number of correctly inferred edges to the number of edges in the correct network (i.e., the number of added edges). We examined observation error levels of 0.1, 0.3, 0.5 and 0.7, for each of which we took the average over 10 trials using randomly generated different parameter values (i.e., a^i_js and $a^i_{j,h}$s), where time-series data were generated as in Section 3.3.2.

The result is shown in Table 3.2, where the accuracy and the average CPU time (user time + sys time) per trial are shown for each case. From this table, the accuracy is high even for large networks in case that the error level is not high. It is also observed that although the CPU time grows rapidly as the size of a network increases, it still remains allowable for networks with 50 genes.

We also carried out a comparative analysis of DPLSQ with two well-known existing tools for genetic network inference, ARACNE [27, 28] and GeneNet [70, 71]. The former serves as a basis for the mutual information and the latter is based on graphical Gaussian models and partial correlations. Computational experiments on ARACNE were carried out under the same environment as that for DPLSQ, whereas those on GeneNet were performed on a PC with Intel Core i7-2600 CPU (3.40 GHz) with 16

Table 3.3: Results on inference of WNT5A network using ARACNE and GeneNet, where the accuracy is shown for each case.

	Method	Observation error level			
		0.1	0.3	0.5	0.7
$n = 10$	ARACNE	0.523	0.523	0.523	0.526
	GeneNet	0.526	0.526	0.533	0.533
$n = 30$	ARACNE	0.332	0.328	0.326	0.326
	GeneNet	0.368	0.380	0.383	0.384
$n = 50$	ARACNE	0.308	0.312	0.310	0.391
	GeneNet	0.313	0.316	0.314	0.316

GB RAM running under the Cygwin on Windows 7 because of the availability of the R platform on which GeneNet works. We employed datasets that were generated in the same way as for DPLSQ and default parameter settings for both tools.

Since both tools output undirected edges to represent their significance values (or their probabilities), we selected the top M edges in the output where M was the number of edges in the original network and regarded $\{v_i, v_j\}$ as a correct edge if either (v_i, v_j) or (v_j, v_i) was included in the edge set of the original network. As shown in Table 3.2, we evaluated the performance in terms of the average accuracy defined as the ratio of the number of correctly inferred edges to the total number of edges in the original network.

As can be seen from Table 3.3, interestingly, both tools show similar performances, each of which does not change much even if the observation error level changes. Readers will probably think that the accuracies shown in Table 3.3 are close to those by random prediction. However, these accuracies were much higher than those obtained by assigning random probabilities to edges, and thus we can mention that these tools outputted meaningful results.

Comparing results from Table 3.2 and Table 3.3, the accuracies by DPLSQ are much higher than those by ARACNE and GeneNet even though both directions of edges are taken into account for ARACNE and GeneNet. However, it should be noted that synthetic time-series data were generated according to the differential equation model on which DPLSQ relies. Therefore, we can only mention that DPLSQ provides good performance if time-series are generated according to appropriate differential equation models. Note also that we can use other differential equation models as long as parameters can be estimated by least-squares fitting.

With regard to computation time, both methods were much faster than DPLSQ. Even for the case of $N = 50$, both ARACNE and GeneNet worked in less than a few

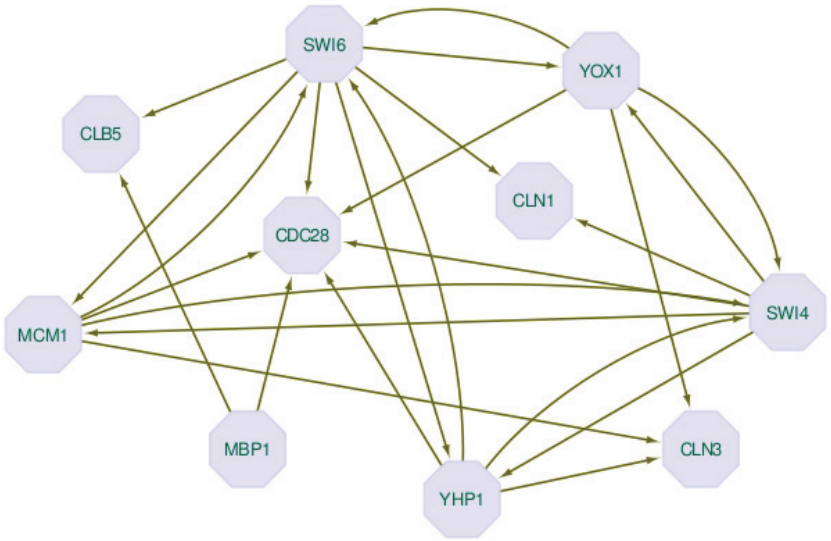


Figure 3.4: Structure of part of yeast cell-cycle network.

seconds per trial. Therefore, DPLSQ does not have merits on practical computation time.

3.3.4 Inference Using Microarray Data

We also tested the performance of DPLSQ for network inference using microarray gene expression data. Since there is no "gold standard" networks for genetic networks and thus the actual underlying genetic network still remains unknown, we did not compare it with the existing methods.

We used a part of the cell-cycle network of *Saccharomyces cerevisiae* extracted from KEGG database [51], shown in Figure 3.4. Although the detailed mechanism underlying cell-cycle still remains unclear, we regard this network as the "correct" answer ("gold standard" network), which may not be true. In this network, each of (MCM1, YOX1, YHP1), (SWI4, SWI6), (CLN3, CDC28), (MBP1, SWI6) is known to constitute a protein complex but we treated them separately and ignored the interactions inside a complex. Because making a protein complex does not necessarily mean a regulation relationship between the corresponding genes.

As for the time-series profiles, we employ the datasets composed of four time-courses (alpha, cdc15, cdc28, elu) in [76] that were measured under four distinct experimental conditions. Since there were several missing values in the time series data, these values were filled by linear interpolation and data on some endpoint time points were discarded because of too many missing values. As the result of normalization, alpha, cdc15, cdc28, and elu datasets consist of expression measurements of 18, 24, 11 and 14 time points, respectively. In order to investigate the relationship between the number of time points and accuracy, we examined four combinations of datasets as summarized in Table 3.4.

Table 3.4: Results on inference of a yeast cell-cycle network by DPLSQ.

Experimental conditions	Accuracy
cdc15	11/25
cdc15+cdc28	8/25
cdc15+cdc28+alpha	8/25
cdc15+cdc28+alpha+elu	11/25

We validated the performance of DPLSQ in terms of the accuracy (i.e., the ratio of the number of correctly inferred edges to the number of added edges), where $H = 3$ and $h = 25$ were used. The result is presented in Table 3.4.

Since the total number of edges in both the original network and the inferred networks is 25 and there exist $9 \times 10 = 90$ possible edges (excluding self-loops), the expected value of corrected edges is roughly estimated as

$$\frac{25}{90} \times 25 = 6.944\ldots, \tag{3.23}$$

if 25 edges are randomly selected and added. Therefore, the results in Table 3.4 indicate that DPLSQ provides better performance than random inference if appropriate datasets are provided (e.g., cdc15 only, or cdc15+cdc28+alpha+elu). It is a bit strange that the accuracies for the first and last datasets are better than those for the second and third datasets because it may usually expected that adding more evidences results in more accurate networks. The possible reason may be that there exist much more measurement noises in time-courses of cdc28 and alpha than those of cdc15 and elu, or some regulation rules that are hidden in Figure 3.4 may be activated under the conditions of cdc28 and/or alpha.

3.3.5 Discussions and Conclusion

This chapter presented a novel method DPLSQ, for completion and inference of gene regulatory networks using dynamic programming and least-squares fitting. The purpose of network completion is to modify a given network so that the resulting network is consistent with the gene expression data. Unlike previous methods, we focused on gene regulatory networks and newly employed a differential equation based model as regulatory rules between genes. The most theoretical difference between DPLSQ and previous methods is that network completion can be done in polynomial time subject to constraints that the maximum indegree of the network is bounded by a constant whereas it is NP-hard in the previous method even if the maximum indegree is bounded by a constant. This difference arises not from the introduction of a least-squares fitting

method but from the assumption that expression levels of all genes are observable.

It should also be noted that most of existing methods for network inference can not guarantee an optimal solution, whereas DPLSQ can, if it is applied to inference of a genetic network with a bounded maximum indegree. Of course, the objective function (i.e., minimizing the sum-of-squared errors) is different from existing ones and thus this property does not necessarily mean that DPLSQ is superior to existing methods in real applications. Indeed, the result using real gene expression data in Section 3.3.4 does not seem to be very good. However, DPLSQ has much room for extensions. For example, least-squares fitting can be replaced by another fitting/regression method (with some regularization term) and the objective function can be replaced by another function as long as it can be computed by sum or product of some error terms. Studies on such improvement might lead to development of better network completion and/or inference methods.

Chapter 4

Exact and Heuristic Methods for Network Completion for Time-Varying Genetic Networks

4.1 Background

Robustness is one of the fundamental features in biological systems that maintain their functions in spite of internal and external perturbations. It has been believed that the alteration in cell function is caused by changes in their genomic programs. Only recently, it is clear that this ability depends on the architecture of genetic networks for temporal changes of underlying processes, such as developmental, immunological response and disease progression [77, 78, 79]. This means that, in order to reveal and understand the dynamic stability specific to biological systems, we need to consider the temporal and structural aspects of gene regulatory networks. These properties will probably be associated with the situation that few genes will invariantly play a role in the cell function, even though the others may exhibit transient behaviors during the specific phase [77].

DNA microarray technology has enabled us to measure the expression levels of a large number of genes simultaneously and to obtain various kinds of biological data, known as gene expression profiles (particularly mRNA expression profiles), CHromatin ImmunoPrecipitation (ChIP)-chip data for transcription binding information, DNA-protein interaction data, and protein-protein interaction data [29, 80, 81]. Thus, analysis of genetic networks based on these observations has proceeded as network inference or reverse-engineering. Traditional methods commonly include such as Boolean networks [5, 6, 67], Bayesian networks [7, 9], differential equations [18, 19, 82, 83] as discussed in Section 1.1.

However, these methods can only describe the single snapshot of the genetic interaction that is, a static "average" network from such an available data. Intrinsically,

static networks are estimated independently of changes in the cellular environmental condition and these models do not provide any temporal information, like the start, end and duration of each stage in developmental processes. The reason is that they assume that genetic networks whose topologies do not change over time, whereas the real gene regulatory network must evolve dynamically their architecture in response to intrinsic and extrinsic cues as described at the beginning of this section.

Needless to say, it is important to develop inference methods which extract the temporal and spatial characterization of genetic networks. Recent approaches often aim to reconstruct time-varying genetic networks from time-series microarray data. For instance, the problem of change point detection in time-series data has been studied in Yoshida *et al.* [84] in which change points in time-series are identified by a dynamical linear model with Markov switching and regimes evolve according to a first-order Markov process. With regard to modeling of dynamic behaviors, Fujita *et al.* [85] proposed a method based on the dynamic autoregressive model. This model extends the vector autoregression (VAR) model, which can be applied to the inference of nonlinear time-dependent biological correlations. Similarly, Robinson and Hartemink [86] presented a model called a non-stationary dynamic Bayesian network, based on dynamic Bayesian networks (DBNs). Lèbrel *et al.* [87] also introduced the autoregressive time-varying (ARTIVA) algorithm for the analysis of time-varying network topologies from time-series generated from different processes. This model adopts a combination of reversible jump Markov chain Monte Carlo (RJMCMC) and DBNs, in which RJMCMC is used for the detection of change time points and the static networks between distinct regimes and DBN is used to represent causal interactions among genes. The method based on the Bayesian network model is studied in Thorne and Stumpf [88], which aims to reconstruct the network structure between distinct segments with a set of hidden states by applying the hierarchical Dirichlet process hidden Markov model [89], including a potentially infinite number of states. The studies in Rasool *et al.* [82] and Khan [77] focused on the smoothed Kalman filtering: in the former, provided a constrained smoothed Kalman filtering method, which is capable of estimating time-varying networks from time-series data, including unobserved and noisy measurements. The dynamics of genetic modules are represented as a linear-state space equation and the observability of linear time-varying systems is defined by imposing sparse constraints in Kalman filters; in the latter, proposed a novel approach of inference of time-varying networks from a limited number of observations defined as a target tracking problem, in which the target is a set of incoming edges to a given gene. Gene expression is assumed to follow the linear dynamics with Kalman filtering for tracking the network connectivities over time and a static gene network is modelled by a standard state-space model. A microarray analysis of *D.melanogaster* was provided that there exist permanent genes and transient genes, which seem to play a role in the overall developmental process and are mainly active during the specific phases. Ahmed

et al. [90] also developed an algorithm called Tesla with machine learning, which can be cast in the form of a convex optimization problem. A gene regulatory network is represented by Markov random fields at arbitrary time intervals. The basic assumption of this method is that individual networks at close time points do not have significant topological differences but have common edges with high probability, in contrast, networks at distant time points are markedly different.

As described above, many studies have been conducted on this theme regardless of the temporal characteristics from microarray observations. However, supposed that genes and proteins in living organisms make up more complicated interactions, any mathematical models have limitations to faithfully reconstruct and cannot provide a realistic view of genetic networks, thus there is not yet a gold standard or established method for inference, even for time-independent networks. One of the possible reasons is that there currently exists an insufficient number of high-quality time-series datasets to reconstruct the dynamics of genetic networks. In other words, it is going to be intrinsically difficult to faithfully reconstruct known/unknown gene regulatory networks based on information derived from noisy and sparse experimental data. Therefore, in order to conduct a realistic analysis, we firstly proposed a new approach for reverse-engineering of time-independent networks from time-series data, called network completion (refer to Chapter 3), whose aim is to make minimum modifications for a given genetic network so as to be consistent with the observed data.

In this chapter, we present two novel methods for completion and inference of time-varying (time-dependent) networks using Dynamic Programming and Least-Squares fitting (DPLSQ): DPLSQ-TV (the preliminary version of DPLSQ-TV is presented in [91]) and DPLSQ-HS, where TV and HS stand for Time-Varying and HeuristicS. DPLSQ-TV is an extension of DPLSQ so as to detect change time points where the network topology change may occur. Since the additions and deletions of edges are basic modifications in network completion, we need to extend DPLSQ so that these operations can be performed at several time points. In DPLSQ-TV, the modified edges and change time points are identified by a novel double dynamic programming method (double DP) algorithm in which inner and outer loops are used for detecting static network structures in each time interval and change points, respectively.

Our methods also allow us to find an optimal solution in polynomial time satisfying the constraint that the maximum indegree (i.e., the maximum number of input genes to a gene) is bounded by a constant. In this regard, DPLSQ-TV provides optimality guarantee in polynomial time, but it has limited use for completion of large-scale networks due to the higher degree polynomials. Therefore, we further propose a heuristic method DPLSQ-HS, to reduce the time complexity by imposing an upper bound on the number of combinations of incoming nodes.

Computational experiments are carried out to test the performance of two methods over synthetic and microarray data during the life-cycle of *D. melanogaster* and the cell-

cycle of *S. cerevisiae*. We also evaluate the effectiveness of two methods by comparing our results with those from the existing method, ARTIVA [87].

4.2 Method

This section presents DPLSQ-TV, which is an exact method for network completion of time-varying networks based on the double DP algorithm. Before going through the details of DPLSQ-TV, let us begin with a few preliminaries on time-varying genetic networks.

4.2.1 Preliminaries

Given a time-varying network, we assume that there exist m measurement time points $(1, 2, \cdots, m)$, which is divided into $B+1$ intervals: $[1, \cdots, c_1 - 1]$, $[c_1, \cdots, c_2 - 1]$, $\cdots$, $[c_B, \cdots, m]$, where B means the number of change points in the given network. For each interval, a different network is associated. Since we assume that the set of genes does not change, only the edge set changes according to the time interval. Let $V = \{v_1, \cdots, v_n\}$ be the set of genes. Let E be the initial set of directed edges (i.e., initial set of gene regulation relationships), and let $E_0, E_1, \cdots, E_B$ be the sets of directed edges (i.e., the output), where E_i denotes the edge set for the i-th interval.

Then, the problem is defined as follows: given an initial network $G(V, E)$ consisting of n genes, N time-series datasets, each of which consists of m time points for n genes and positive integers h, w, and B, infer B change points (i.e., $c_1, c_2, \cdots, c_B$) and complete or infer the initial network $G(V, E)$ by adding h edges and deleting w edges in total such that the total least-squared error is minimized. This results in the set of edges $E_0, E_1, \cdots, E_B$ at the corresponding time intervals (see Figure 4.1). It is to be noted that if we start with an empty set of edges (i.e., $E = \emptyset$), the problem corresponds to the inference of a time-varying network.

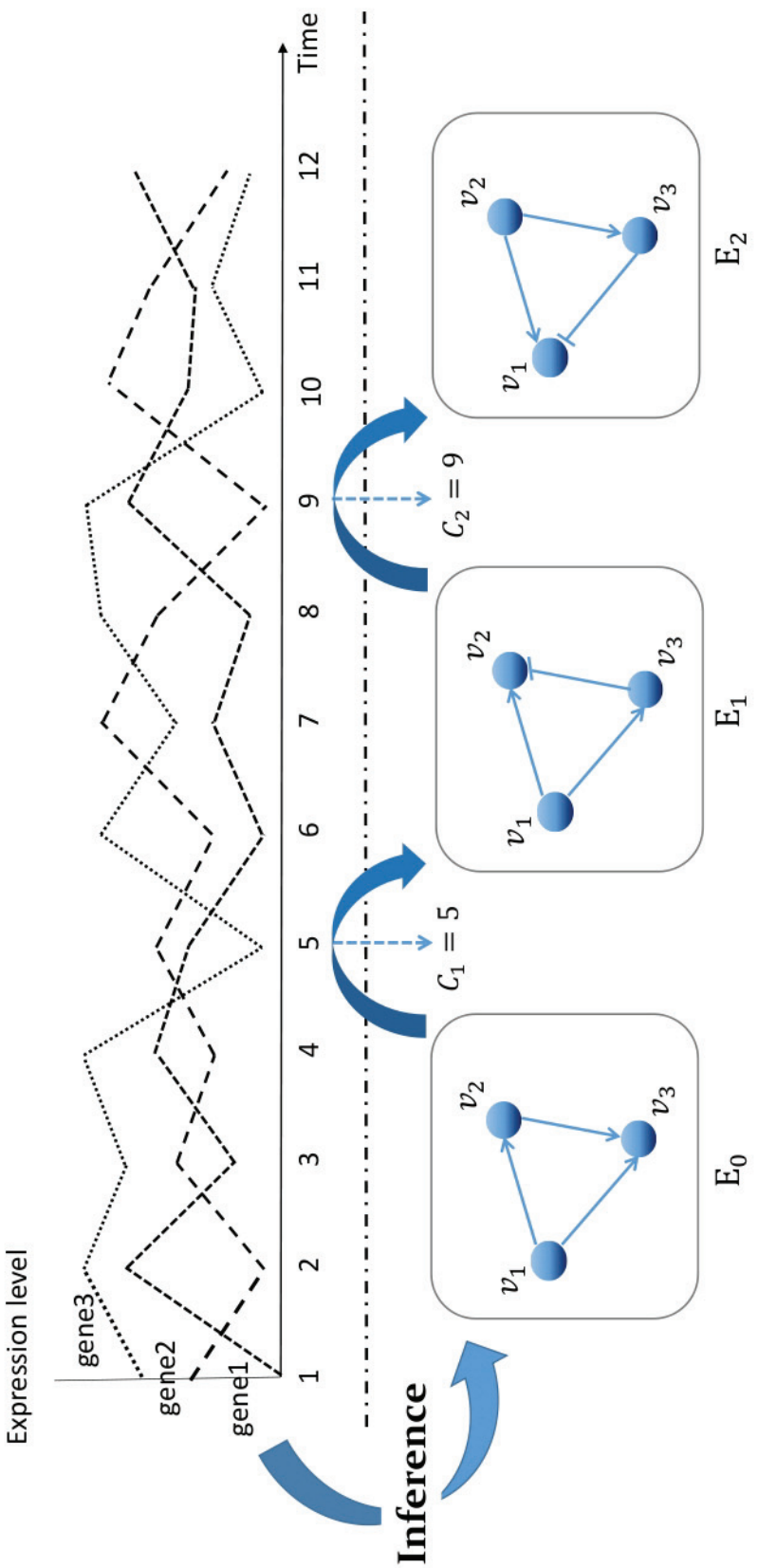


Figure 4.1: Overview of the network inference for time-varying genetic networks. Inference (i.e., completion starting with the null network) of time-varying structure during each time interval. This example corresponds to the case of $N = 1$, $n = 3$, $B = 2$, $m = 12$. The change points are $c_1 = 5$ and $c_2 = 9$.

4.2.2 Model of Differential Equation and Estimation of Parameters

To model the regulatory rules between genes, we assume that dynamics of each node v_i $(i = 1, \cdots, n)$ is determined by a following differential equation:

$$\frac{dx_i}{dt} = a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j} + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j} x_{i_h} + b^i \omega, \tag{4.1}$$

where $v_{i_1}, \cdots, v_{i_d}$ are incoming nodes to v_i, x_i corresponds to the expression value of node v_i, and ω denotes a random noise. The second and third terms of the right-hand side of the equation represent linear and nonlinear effects to node v_i, respectively (see Figure 3.2), where positive a_j^i or $a_{j,h}^i$ corresponds to an activation effect and negative a_j^i or $a_{j,h}^i$ corresponds to an inhibition effect. This model is an extension of the linear differential equation model [29]. It is also a variant of the recurrent neural network model [95], although the sigmoid function is replaced here by an identify function and non-linear terms representing cooperating regulations are added instead.

In practice, we replace the derivative of Equation (4.1) by the difference, and ignore the noise term as follows:

$$x_i(t+1) = x_i(t) + \left(a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j}(t) + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j}(t) x_{i_h}(t) \right). \tag{4.2}$$

This kind of discretization is also employed for linear and recurrent neural network models [29, 95]. We assume that time-series data $y_i(t)$s, which correspond to $x_i(t)$s in Equation (4.2), are given for $t = 0, 1, \ldots, m$, where we distinguish an observed expression value $y_i(t)$ from an expression value $x_i(t)$ in the mathematical model Equation (4.2). Then, the parameters a_j^is and $a_{j,h}^i$s are estimated from these time-series data by minimizing the following objective function (i.e., the sum-of-squared errors) for each node v_i:

$$\sum_{t=1}^{m} \left| y_i(t+1) - \left[y_i(t) + \left(a_0^i + \sum_{j=1}^{d} a_j^i y_{i_j}(t) + \sum_{1 \le j < h \le d} a_{j,h}^i y_{i_j}(t) y_{i_h}(t) \right) \right] \right|^2. \tag{4.3}$$

It should be noted that $y_i(t)$ is the observed expression value of gene v_i at time t, and $v_{i_1}, v_{i_2}, \ldots, v_{i_h}$ are tentative incoming nodes to node v_i. Incoming nodes to each node are determined so that the sum of these values for all nodes is minimized under the constraint that the total number of edges is equal to the specified number. In order to minimize the sum of least-squared errors for all genes along with determining the incoming nodes and corresponding parameters, DP is applied. Readers are referred to Chapter 3 for the details of DPLSQ.

4.2.3 Completion by Addition of Edges

In this subsection, we present our proposed method for network completion of time-varying networks by the addition of edges, and extend this to a general case (i.e., network completion by the addition and deletion of edges) in the following subsection. For simplicity, we assume $N = 1$, where we can extend the method to the case of $N > 1$ by changing the definition of $S^i_{\left\{v_{i_1}, v_{i_2}, \cdots, v_{i_d}\right\}}[p, q]$ only.

Given the set of nodes (i.e., the set of genes) V and the set of initial edges E, we denoted by $\{v_{i_1}, \cdots, v_{i_d}\}$ the set of incoming nodes to v_i and define the sum-of-squared error for v_i during the time period between p and q as

$$S^i_{\left\{v_{i_1}, v_{i_2}, \cdots, v_{i_d}\right\}}[p, q] = \min_{a^i_0,\, a^i_j,\, a^i_{j,h}} \sum_{t=p}^{q-1} \left| y_i(t+1) - \left[y_i(t) + \left(a^i_0 + \sum_{j=1}^{d} a^i_j y_{i_j}(t) + \sum_{1 \le j < h \le d} a^i_{j,h} y_{i_j}(t)\, y_{i_h}(t) \right) \right] \right|^2, \tag{4.4}$$

where $y_i(t)$ denotes the observed expression level for gene v_i at time t. The parameters (i.e., a^i_0, a^i_j, $a^i_{j,h}$) needed to attain this minimum value can be computed by a standard least-squares fitting method.

Because network completion is considered to involve the addition of edges, let $e^-(v_j) = \{v_{j_1}, \cdots, v_{j_d}\}$ denote the set of initial incoming nodes to v_j in a given network. Let $\sigma^+_{h_j, j}[p, q]$ denote the minimum sum-of-squared error when adding h_j edges to the j-th node during the time from p to q, which is formally defined as Equation (4.5),

$$\sigma^+_{h_j, j}[p, q] = \min_{j_1, j_2, \cdots, j_{h_j}} \left\{ S^j_{e^-(v_j) \cup \left\{v_{j_1}, v_{j_2}, \cdots, v_{j_{h_j}}\right\}}[p, q] \right\}, \tag{4.5}$$

where each v_{j_l} must be selected from $V - v_j - e^-(v_j)$. In order to avoid combinatorial explosion, we constrain the maximum h_j to be a small constant, H, and let $\sigma^+_{h_j, j}[p, q] = +\infty$ for $h_j > H$ or $h_j + |e^-(v_j)| \ge n$.

Then, the problem is stated as

$$\min_{\substack{c_1 < c_2 < \cdots < c_B \\ h^0 + h^1 + \cdots + h^B = h}} \left\{ \sum_{i=0}^{B} \min_{h_1 + h_2 + \cdots + h_n = h^i} \left[\sum_{j=1}^{n} \sigma^+_{h_j, j}[c_i, c_{i+1} - 1] \right] \right\}, \tag{4.6}$$

where $c_0 = 1$ and $c_{B+1} - 1 = m$.

Here, we define $D^+[h,i,p,q]$ as

$$D^+[h,i,p,q] = \min_{h_1+h_2+\cdots+h_i=h} \left\{ \sum_{j=1}^{i} \sigma^+_{h_j,j}[p,q] \right\}. \tag{4.7}$$

The elements of $D^+[h,j,p,q]$ can be computed by the following DP algorithm:

$$\begin{aligned} D^+[h,1,p,q] &= \sigma^+_{h,1}[p,q], \\ D^+[h,j+1,p,q] &= \min_{h'+h''=h} \left\{ D^+[h',j] + \sigma^+_{h'',j+1}[p,q] \right\}. \end{aligned} \tag{4.8}$$

Finally, $D^+[h,n,p,q]$ is determined uniquely regardless of the ordering of nodes in the network. The correctness of this DP algorithm can be proved as follows:

$$\begin{aligned} &\min_{h_1+h_2+\cdots+h_n=h} \left\{ \sum_{j=1}^{n} \sigma^+_{h_j,j}[p,q] \right\} \\ &= \min_{h'+h''=h} \left\{ \min_{h_1+h_2+\cdots+h_{n-1}=h'} \sum_{j=1}^{n-1} \sigma^+_{h_j,j}[p,q] + \sigma^+_{h'',n}[p,q] \right\} \\ &= \min_{h'+h''=h} \left\{ D^+[h',\, n-1,p,q] + \sigma^+_{h'',n}[p,q] \right\}. \end{aligned} \tag{4.9}$$

Next we define $E^+[h,b,q]$ as

$$E^+[h,b,q] = \min_{\substack{c_1<c_2<\cdots<c_b \\ h^0+h^1+\cdots+h^b=h}} \left\{ \sum_{i=0}^{b} \min_{h_1+h_2+\cdots+h_n=h^i} \left[\sum_{j=1}^{n} \sigma^+_{h_j,j}[c_i, c_{i+1}-1] \right] \right\}, \tag{4.10}$$

where b is the number of change points and $c_{b+1}-1=q$. $E^+[h,b,q]$ can be computed by the following DP algorithm:

$$\begin{aligned} E^+[h,0,q] &= D^+[h,n,1,q], \\ E^+[h,b,q] &= \min_{\substack{p\in\{1,\cdots,q-1\} \\ h'+h''=h}} \left\{ E^+[h',b-1,p] + D^+[h'',n,p,q] \right\}. \end{aligned} \tag{4.11}$$

The introduction of $E^+[h,b,q]$ and the corresponding procedure based on the double dynamic programming technique are the methodologically novel points of this work, compared with DPLSQ.

The correctness of this DP algorithm can be proved as follows:

$$\begin{aligned}
&\min_{\substack{c_1<c_2<\cdots<c_b \\ h^0+h^1+\cdots+h^b=h}} \left\{ \sum_{i=0}^{b} \min_{h_1+h_2+\cdots+h_n=h^i} \left[\sum_{j=1}^{n} \sigma^{+}_{h_j,j}\left[c_i, c_{i+1}\text{-}1\right] \right] \right\} \\
&= \min_{\substack{c_{b\text{-}1}<c_b \\ h'+h''=h}} \left\{ \min_{\substack{c_1<\cdots<c_{b\text{-}1} \\ h^0+\cdots+h^{b-1}=h'}} \left\{ \sum_{i=0}^{b-1} \min_{h_1+h_2+\cdots+h_n=h^i} \left[\sum_{j=1}^{n} \sigma^{+}_{h_j,j}\left[c_i, c_{i+1}\text{-}1\right] \right] \right\} \right. \\
&\qquad \left. + \min_{h_1+h_2+\cdots+h_n=h''} \left[\sum_{j=1}^{n} \sigma^{+}_{h_j,j}\left[p,q\right] \right] \right\} \\
&= \min_{\substack{p\in\{1,\cdots,q-1\} \\ h'+h''=h}} \left\{ E^{+}\left[h', b-1, p\right] + D^{+}\left[h'', n, p, q\right] \right\},
\end{aligned} \tag{4.12}$$

where $c_b = p$, $c_{b+1} - 1 = q$.

4.2.4 Completion by Addition and Deletion of Edges

The above DP procedure can be developed for the deletion of edges and for the addition and deletion of edges as in DPLSQ. Since the former case is the subcase of the latter one, we describe here only the latter one (addition and deletion of edges).

Let $\sigma_{h_j,w_j,j}[p,q]$ denote the minimum sum-of-squared error for the time interval between p and q when adding h_j edges to $e^{-}(v_j)$ and deleting w_j edges from $e^{-}(v_j)$ as below, where the added and deleted edges must be disjointed.

$$\sigma_{h_j,w_j,j}\left[p,q\right] = \min_{\substack{j_1,j_2,\ldots,j_{h_j} \\ j'_1,j'_2,\ldots,j'_{w_j}}} \left\{ S^{j}_{e^{-}(v_j)-\{v_{j'_1},v_{j'_2},\ldots,v_{j'_{w_j}}\}\cup\{v_{j_1},v_{j_2},\ldots,v_{j_{h_j}}\}}\left[p,q\right] \right\}, \tag{4.13}$$

where $\{v_{j'_1}, v_{j'_2}, \ldots, v_{j'_{w_j}}\}$ is the set of deleted edges from $e^{-}(v_j)$. We constrain the maximum h_j and w_j to the small constants H and W and let $\sigma_{h_j,w_j,j}[p,q] = +\infty$ if $h_j > H$, $w_j > W$, $h_j - w_j + |e^{-}(v_j)| \geq n$, or $h_j - w_j + |e^{-}(v_j)| < 0$ hold. Then, the problem is stated as

$$\min_{\substack{c_1<c_2<\cdots<c_B \\ h^0+h^1+\cdots+h^B=h \\ w^0+w^1+\cdots+w^B=w}} \left\{ \sum_{i=0}^{B} \min_{\substack{h_1+h_2+\cdots+h_n=h^i \\ w_1+w_2+\cdots+w_n=w^i}} \left[\sum_{j=1}^{n} \sigma_{h_j,w_j,j}\left[c_i, c_{i+1}-1\right] \right] \right\}. \tag{4.14}$$

Here, we define $D[h, w, i, p, q]$ as

$$D[h, w, i, p, q] = \min_{\substack{h_1+h_2+\cdots+h_i=h \\ w_1+w_2+\cdots+w_i=w}} \left\{ \sum_{j=1}^{i} \sigma_{h_j, w_j, j}[p, q] \right\}. \tag{4.15}$$

As in the previous subsection, $D[h, w, j, p, q]$ can be computed by

$$\begin{aligned} D[h, w, 1, p, q] &= \sigma_{h, w, 1}[p, q], \\ D[h, w, j+1, p, q] &= \min_{\substack{h'+h''=h \\ w'+w''=w}} \{D[h', w', j, p, q] + \sigma_{h'', w'', j+1}[p, q]\}. \end{aligned} \tag{4.16}$$

Next, we define $E[h, w, b, q]$ as

$$E[h, w, b, q] = \min_{\substack{c_1<c_2<\cdots<c_b \\ h^0+h^1+\cdots+h^b=h \\ w^0+w^1+\cdots+w^b=w}} \left\{ \sum_{i=0}^{b} \min_{\substack{h_1+h_2+\cdots+h_n=h^i \\ w_1+w_2+\cdots+w_n=w^i}} \left[\sum_{j=1}^{n} \sigma_{h_j, w_j, j}[c_i, c_{i+1}-1] \right] \right\}. \tag{4.17}$$

$E[h, w, b, q]$ can be computed by the following DP algorithm:

$$\begin{aligned} E[h, w, 0, q] &= D[h, w, n, 1, q], \\ E[h, w, b, q] &= \min_{\substack{p \in \{1, \cdots, q-1\} \\ h'+h''=h \\ w'+w''=w}} \{E[h', w', b-1, p] + D[h'', w'', n, p, q]\}. \end{aligned} \tag{4.18}$$

4.2.5 Time Complexity Analysis

In this subsection, we present computational complexity analysis of DPLSQ-TV. For the assessment of time complexity of DPLSQ-TV, we estimate the execution time for network completion by adding and deleting of edges. In general, it is known that least-squares fitting for a linear system can be done in $O(mp^2 + p^3)$ time complexity as indicated in Section 2.2.4, assuming that m is the sample size and p is the number of predictor variables. By introducing constraints on our model with bounded indegree, DPLSQ-TV enables us to decrease the time complexity from $O(mp^2 + p^3)$ to $O(m)$ (refer to Section 3.2.6). In terms of the computation of $\sigma_{h_j, w_j, j}[p, q]$s, the computation of $\sigma_{h_j, w_j, j}$ takes $O\left(mn^{H+1}\right)$ time as discussed in Section 3.2.6, thereby providing the time complexity for $\sigma_{h_j, w_j, j}[p, q]$s is $O\left(m^3 n^{H+1}\right)$, because p and q are $O(m)$.

Next, we analyze the time complexity required for computing $D[h, w, i, p, q]$s. In this computation, we must note that the size of table $D[h, w, i, p, q]$ is $O(m^2 n^3)$. Furthermore, in order to compute the minimum value for each entry in the DP procedure, we need to examine $(H+1)(W+1)$ combinations, which is $O(1)$. Hence, the time

complexity for $D[h, w, i, p, q]$s is $O(m^2 n^3)$.

Also, we analyze the time complexity required for computing $E[h, w, b, q]$s. Note that the size of table $E[h, w, b, q]$ is $O(mn^2)$, where we assume that b is a constant. Since the number of combinations for computing the minimum value using DP is $O(mn)$ per entry, the computation time required for computing $E[h, w, b, q]$s is $O(m^2 n^3)$. Consequently the overall computational complexity can be estimated as below:

$$O\left(m^3 n^{H+1} + m^2 n^3\right). \tag{4.19}$$

It must be noted that, if we use N time-series datasets, each of which consists of m time points, the total time complexity becomes $O(Nm^3 n^{H+1} + m^2 n^3)$. This complexity is not small, however it is allowable in practice if $H \leq 2$ and n and m are not too large. Indeed, as shown in Section 4.4.3, DPLSQ-TV works for the completion and inference of time-varying networks with a few tens of genes if $H = 2$.

4.3 Heuristic Method

Although our previous algorithm for exact completion DPLSQ-TV, is guaranteed to find an optimal solution in polynomial time, the higher degree polynomial prevents this method from being applied to the completion of large-scale networks. The reason why DPLSQ-TV requires a lot of CPU time is that least-squared errors are calculated for each node by considering all possible combinations of incoming nodes and taking the minimum value of them. Therefore, we propose a heuristic algorithm DPLSQ-HS, to significantly improve the computational efficiency by relaxing the optimality condition. In order to improve the computational efficiency, we introduce an upper limit on the number of combinations of incoming nodes. Although DPLSQ-HS does not guarantee an optimal solution, it allows us to reduce the computation time of computing of the minimum least-squares in the case of adding edges. This computational procedure will be schematically illustrated in the next subsection. The detailed description of DPLSQ-HS algorithm will be presented in Section 4.3.2, and we will discuss the time complexity of DPLSQ-HS in Section 4.3.3.

4.3.1 Schematic Illustrations of Computational Procedures

Although DPLSQ-HS can be applied to completion by adding and deleting of edges, we schematically show the procedure only for additions of edges, because we impose an upper limit only on the number of adding edges. In particular, we have developed DPLSQ-HS, which contributes to reducing the time complexity, by imposing the constraint on the number of combinations of incoming edges to each node. In Figure 4.2, the diagram indicates that, for each node v_i, we maintain M combinations of h incoming nodes with M lowest errors at the h-th step. Let S_i^h denote the set of M combinations computed at the h-th step for v_i. At the h-th step, for each combination $\{v_{i_1}, \ldots, v_{i_{h-1}}\} \in S_i^{h-1}$ where $i_1 < i_2 < \cdots < i_{h-1}$, we calculate the least-squared error for each v_j such that $j > i_{h-1}$ is the h-th incoming node to v_i. The calculated least-squared errors are sorted in ascending order, the top M values are selected, and the corresponding combinations are stored in S_i^h.

4.3.2 Description of Algorithm

In the following, we give a detailed description of the algorithm to compute $\sigma^+_{h,\,i}[p, q]$ in DPLSQ-HS, where $\sigma^+_{h,\,i}[p, q]$ does not necessarily mean the minimum value, and the meaning of 'Step' is different from that in Section 4.3.1.

Step 1: For each period $[p, q]$, repeat Steps 2-6.

Step 2: Let $S_i^0 = \{\emptyset\}$ for all $i = 1, \ldots, n$.

Step 3: For $i = 1$ to n do Steps 4-7.

Step 4: Repeat Steps 5-7 for each node v_i from $h = 1$ to $h = H$.

Step 5: For each combination $\{v_{i_1}, \ldots, v_{i_{h-1}}\} \in S_i^{h-1}$ and each node v_j such that $j > i_{h-1}$ ($j > 0$ if $h = 1$), calculate the least-squared error for the h edge set $\{(v_{i_1}, v_i), \ldots, (v_{i_{h-1}}, v_i), (v_j, v_i)\}$.

Step 6: Sort the obtained least-squared errors in ascending order and select the top M combinations, which are stored in S_i^h.

Step 7: Let $\sigma^+_{h,\,i}[p, q]$ be the minimum least-squared error among these top M combinations.

The other parts of the algorithm are the same as in DPLSQ-TV.

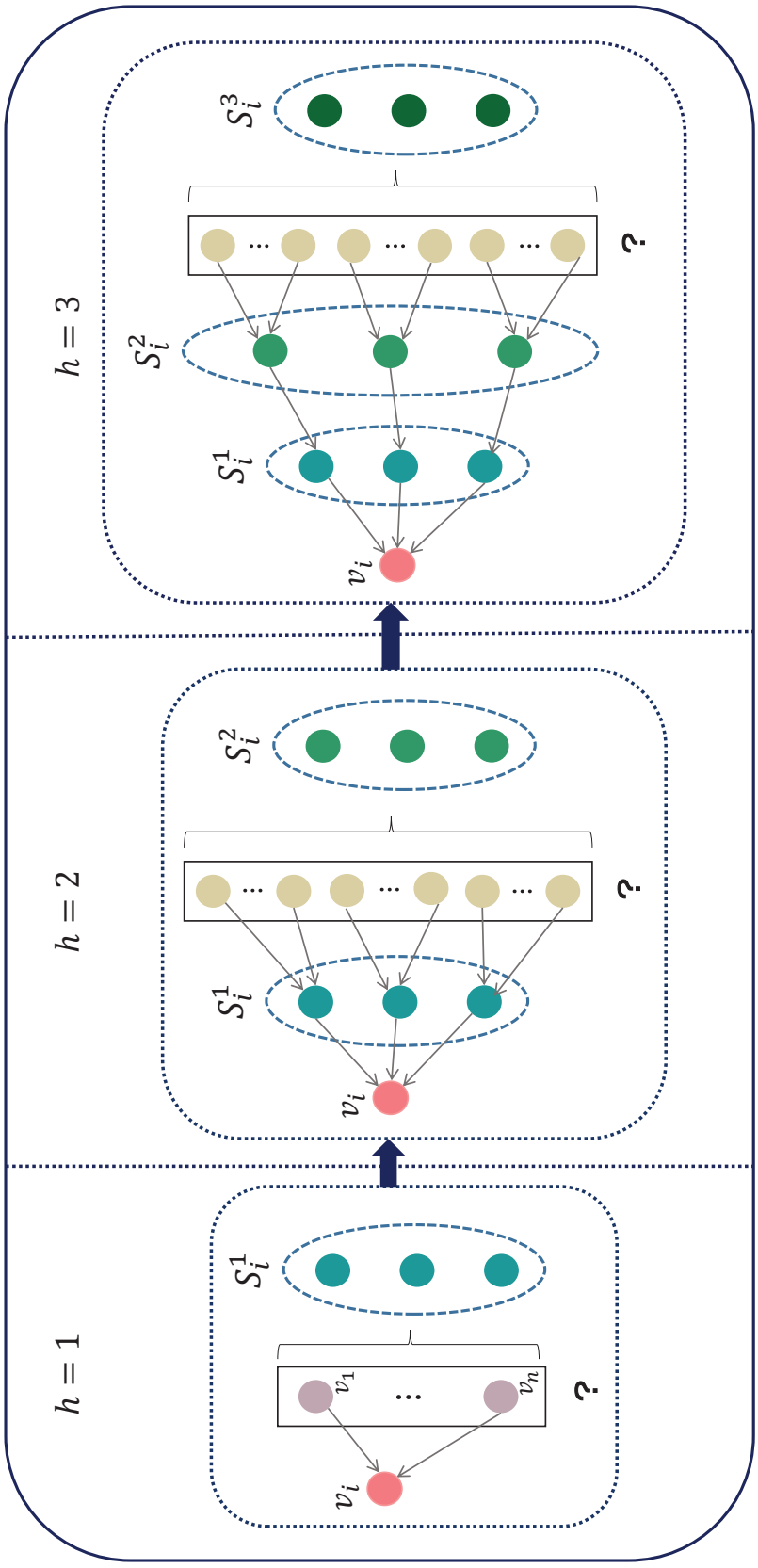


Figure 4.2: Schematic illustrations of the definition of the top M combinations. This is an example that the case of $M = 3$ and $h \leq 3$. Let S_i^h denote the set of M combinations computed at the h-th step. At the h-th step, for each combination $\{v_{i_1}, \ldots, v_{i_{h-1}}\} \in S_i^{h-1}$, we calculate the least-squared error for each v_j such that $j > i_{h-1}$ as a h-th incoming node to v_i.

4.3.3 Time Complexity Analysis

In this subsection, we analyze the time complexity of DPLSQ-HS. Since DPLSQ-HS can be applied to additions and deletions of edges, we consider the time complexity of completion for both adding and deleting edges.

In our proposed method, we assume that the numbers of adding and deleting edges in a given network are bounded by constants H and W, respectively. In this case, the time complexity for least-squares fitting can be estimated as $O(m)$ (refer to Section 2.2.4).

With regard to the time complexity of computing $\sigma_{h_j,w_j,j}[p,q]$, we assume that additions of edges are operated only in the case of adding edges to the nodes with respect to the top M of the sorted list. Therefore, the number of combinations of addition of h_j edges, which is bounded by a constant H, is $O(Mn)$. It is well known that the sorting of n data can be done in $O(n \log n)$ time. Based on such assumptions, the total computation time required for the computation of $\sigma_{h_j,w_j,j}[p,q]$ is $O(mn \log n)$ [93] since the $O(Mn)$ factor can be regarded as a constant. Therefore, the time complexity for $\sigma_{h_j,w_j,j}[p,q]$ is $O(m^3 n \log n)$, because p and q are $O(m)$.

Furthermore, for the time complexity required for computing $D[h,w,i,p,q]$s and $E[h,w,b,q]$s, the calculation process is the same as that in DPLSQ-TV. Therefore, the computation time for both $D[h,w,i,p,q]$s and $E[h,w,b,q]$s are $O(m^2 n^3)$ as described in Section 4.2.5. Hence, the overall time complexity of DPLSQ-HS is

$$O\left(m^3 n \log n + m^2 n^3\right). \tag{4.20}$$

If we use N time-series datasets, each of which consists of m points, the total time complexity becomes $O(Nm^3 \log n + m^2 n^3)$. DPLSQ-HS requires less time complexity than DPLSQ-TV, because $O(m^3 n \log n)$ is much smaller than $O\left(m^3 n^{H+1}\right)$. Indeed, as shown in Section 4.2.5, DPLSQ-HS is much faster than DPLSQ-TV in practice.

4.4 Results and Discussion

4.4.1 Availability

We validated the performance of proposed methods using both synthetic data and microarray expression data. All experiments were performed on a PC with Intel Core 2 Quad CPU Q9650 (3.00 GHz) with 186 GB RAM running under the Linux kernel 2.6.32.23-170.fc12. We employed the liblsq library (http://www2.nict.go.jp/aeri/sts/stmg/K5/VSSP/install_lsq.html) for a least-squares fitting method.

4.4.2 Completion Using Synthetic Data

In order to evaluate the potential effectiveness of DPLSQ-TV and DPLSQ-HS, we start with network completion for time-varying networks using synthetic data. We test our performance in terms of detecting change time points that the sum-of-squared errors are minimized. We employed a randomly generated network consisting 10 genes as an initial network G, and three different networks G_1, G_2 and G_3 generated by randomly adding and deleting edges from the initial network. In this method, for each node v_i with d input nodes, we considered the following model:

$$x_i(t+1) = x_i(t) + \left(a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j} + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j}(t) x_{i_h}(t) + b_i \omega \right), \tag{4.21}$$

where a_j^is and $a_{j,h}^i$s are constants selected uniformly at random from $[-0.5, 0.5]$ and $[-0.05, 0.05]$, respectively. The reason why the domain of $a_{j,h}^i$s is smaller than that for a_j^is is that non-linear terms are not considered as strong as linear terms. It should also be noted that $b_i\omega$ is a stochastic term, where b_i is a constant (we used $b_i = 0.05$) and ω is random noise taken uniformly at random from $[-1, 1]$. For the artificial generation of the observed data $y_i(t)$, we used

$$y_i(t) = x_i(t) + o^i \epsilon, \tag{4.22}$$

where o^i is a constant denoting the level of observation errors and ϵ is random noise taken uniformly at random from $[0.05, -0.05]$.

As for the time-series data, we generated an original dataset with 30 time points including two change points $c_1 = 10$, $c_2 = 20$, which is generated by merging three datasets for G_1, G_2 and G_3. Since the use of time-series data beginning from only one set of initial values easily resulted in rank deficiency, we generated additional time-series data beginning from 40 sets of initial values that were obtained by slightly perturbing the original data.

Under the above model, we conducted computational experiments by DPLSQ-TV and DPLSQ-HS in which the initial network G was modified by randomly adding h_0 edges and deleting w_0 edges per node, resulting in G_1, G_2 and G_3 using the default values of $h_0 \le 2$, $w_0 \le 2$ and the default values of $H = W = 2$. We evaluated the performance of two methods in terms of the accuracy of modified edges, the time point errors for time intervals, and the execution time for completion (CPU time). Furthermore, in order to examine how CPU time changes as the size of the network grows, we generated networks with 20, 30, 40 and 60 genes by making 2, 3, 4 and 6 copies of the original networks. We examined observation error levels of 0.05, 0.1, 0.3 and 0.5, for each of which we took the average time point errors, accuracies, and CPU

time over 10 random modifications.

The accuracy is defined as follows:

$$\frac{h + w + \sum_{i=0}^{B} (|E_i \cap E'_i| - |E_i|)}{h + w}, \tag{4.23}$$

where E_i and E'_i $(i = 0, 1, \cdots, B)$ are the sets of edges in the original network and the completed network in each time interval, respectively. This value is 1 if all the added and deleted edges are correct and 0 if none of the added and deleted edges is correct. If we regard a correctly (resp., incorrectly) added or deleted edge as a true (resp., false) positive, $\sum_{i=0}^{B}(|E_i| - |E_i \cap E'_i|)$ corresponds to the number of false positives and $h + w + \sum_{i=0}^{B}(|E_i \cap E'_i| - |E_i|)$ corresponds to the number of true positives. The time point error means the average distance between the actual observed and estimated values for change time points and is defined as

$$\frac{1}{B}\sum_{i=1}^{B} |c_i - c'_i|, \tag{4.24}$$

where c'_i $(i = 1, 2, \cdots, B)$ are estimated change points. As for the execution time, we show the average CPU time.

The results of the two methods are summarized in Table 4.1. As can be seen from this table, the change time point errors are almost zero regardless of the network size and the level of observation errors. In addition, we are able to observe that the CPU time using DPLSQ-TV rapidly increases as the size of the network grows, but on the other hand, the CPU time by DPLSQ-HS increases gradually in case that the size of the network is more than 20. In particular, the DPLSQ-HS algorithm is within $4-5$ times and $8-10$ times faster than the DPLSQ-TV algorithm in case of 40 and 60 genes, respectively, while maintaining reasonably good accuracy. Hence, these results suggest that DPLSQ-TV and DPLSQ-HS can accurately detect change time points and that they can complete given networks by modifying the edges with relatively good accuracy if the error levels are not very large.

It must be noted that DPLSQ-HS worked reasonably fast even for $n = 60$ although DPLSQ-TV took about 40000 seconds per execution. However, the accuracies on DPLSQ-HS became around 0.3 even if the observation error level was low (i.e., $o^i = 0.05/0.1$). Therefore, DPLSQ-HS has limited applicability with respect to the accuracy of modified edges, although it may still be useful for networks with $n = 60$ if the purpose is to detect change time points.

Table 4.1: Result on completion with synthetic data.

(a) Using DPLSQ-TV

		Observation error level			
		0.05	0.1	0.3	0.5
$n = 10$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.464	0.474	0.401	0.329
	CPU time (sec.)	102.211	121.340	135.762	119.680
$n = 20$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.347	0.332	0.326	0.324
	CPU time (sec.)	1721.682	1392.793	1427.215	1380.156
$n = 30$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.325	0.317	0.248	0.245
	CPU time (sec.)	5263.687	4389.434	4114.544	4174.860
$n = 40$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.366	0.315	0.246	0.253
	CPU time (sec.)	10993.607	10658.169	10631.024	10702.756
$n = 60$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.330	0.329	0.287	0.280
	CPU time (sec.)	59413.264	44507.814	36793.397	35799.802

(b) Using DPLSQ-HS

		Observation error level			
		0.05	0.1	0.3	0.5
$n = 10$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.363	0.464	0.393	0.308
	CPU time (sec.)	71.399	86.104	94.550	84.207
$n = 20$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.338	0.286	0.289	0.275
	CPU time (sec.)	463.798	456.222	478.047	474.031
$n = 30$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.343	0.279	0.211	0.239
	CPU time (sec.)	795.251	1277.968	1323.173	1308.003
$n = 40$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.342	0.278	0.224	0.272
	CPU time (sec.)	1865.118	2149.137	2224.621	2315.528
$n = 60$	Time point error	0.00	0.00	0.00	0.00
	Accuracy	0.341	0.334	0.249	0.330
	CPU time (sec.)	3947.643	4397.677	4547.498	4384.758

Table 4.2: Validation of stability of proposed methods.

	DPLSQ-TV	DPLSQ-HS
$o^i = 0.3$	0.01325	0.01835
$o^i = 0.5$	0.02842	0.03430

Since DPLSQ-HS is a heuristic method, the results may be greatly influenced by types of data. Therefore, we validated the algorithmic stability of DPLSQ-HS by comparing the variance of the accuracy with that for DPLSQ-TV. This validation was performed on examining variance of accuracies of 10 trials, where $n = 20$ and the results are presented in Table 4.2. From the table, we can see that DPLSQ-HS was nearly as stable as DPLSQ-TV becasue the variances of DPLSQ-HS is almost equal to those of DPLSQ-TV. This result suggests that DPLSQ-HS also has some algorithmic stability.

In addition, we carried out another experiments with varying parameters (one experiment per parameter) in order to examine the relationship between the number of change points B and the maximum number of added and deleted edges for each node, H and W on the least-squared errors. The resulting sum-of-squared errors (i.e., $E[\ldots]$s) for DPLSQ-TV are 3.105, 3.321, 3.518, 2.451, and 2.651 for $(B, H, W) = (2, 1, 1)$, $(3, 1, 1)$, $(4, 1, 1)$, $(2, 2, 2)$, and $(2, 4, 2)$, respectively. From this result, the use of larger H, W resulted in smaller least-squared errors. It is reasonable that more parameters resulted in better least-squares fitting. However, use of larger B did not result in smaller least-squared errors. It may be because addition of unnecessary change points increases the error if an enough number of edges are not added. It must be noted that although the least-squared errors are reduced, use of larger H, W is not always appropriate because it needs much longer CPU time and may cause overfitting.

Finally, we also compared our results with those obtained by the ARTIVA algorithm [87], which is an accessible tool for the inference of time-varying networks. This method is based on a combination of DBN and RJMCMC sampling, where RJMCMC is used to approximate the posterior distribution and DBN is used to infer simultaneously change points and resulting network structures. We applied ARTIVA to the synthetic datasets that were generated in the same way as for our proposed methods. We set the default values of parameters and evaluated the results by detecting change points. As seen in Figure 4.3, ARTIVA can only detect one change point among three change points regardless of the level of the observation error. It is noted that ARTIVA does not uniquely detect change points but output probabilities of change points.

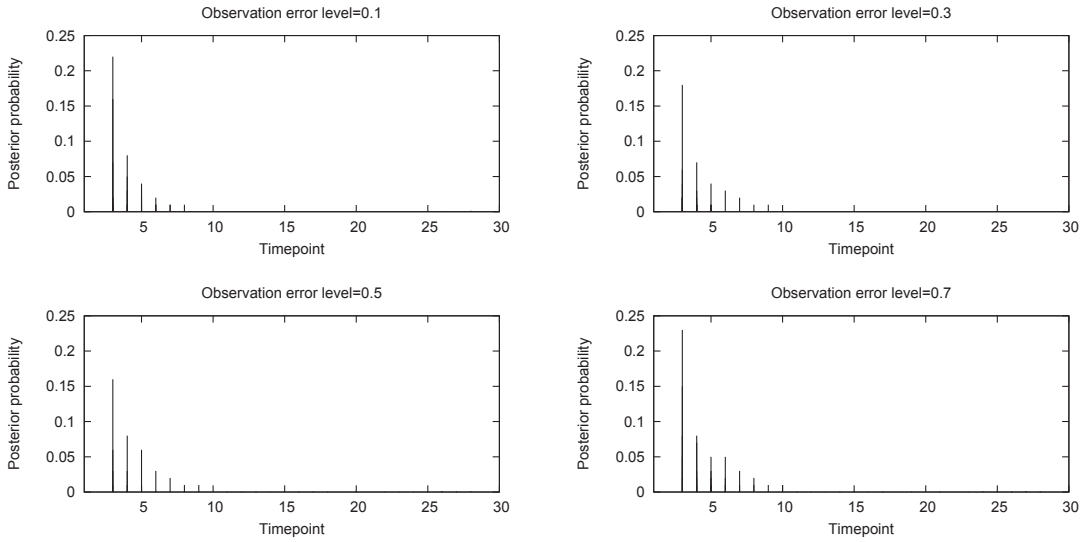


Figure 4.3: Comparative analysis with the ARTIVA algorithm for the detection of change points in the synthetic time-series data.

4.4.3 Inference Using Real Data

We tested the performance of DPLSQ-TV and DPLSQ-HS using two types of microarray data, and also compared our results with those obtained using the ARTIVA algorithm. We applied our methods to two real gene expression datasets, measured during the cell-cycle of *S. cerevisiae* and the life-cycle of *D. melanogaster*.

The first microarray dataset is the expression time-series collected by Spellman *et al.* [76]. In this comparison, we selected 10 genes associated with part of the cell-cycle network extracted from the KEGG database [51] shown in Figure 3.4. As for time-series data, we combined and employed four sets of time-series data (alpha, cdc15, cdc28, and elu) in [76] that were obtained in four different experiments. We adopted the datasets of 10 genes with 71 time points including three change time points. Since there were several values that were far away from the mean in the cdc15 dataset, thereby discarding them. As a result, the alpha, cdc15, cdc28, and elu datasets consist of 18, 23, 17, and 13 time points, respectively.

The second microarray dataset is the time-series taken from Arbeitman *et al.* [94]. It contains the expression levels of 4028 genes with 67 successive time points spanning four distinct stages: embryonic (31 time points), larval (10 time points), pupal (18 time points), and adulthood (8 time points) in the *D. melanogaster* life-cycle. We selected 30 genes (TFs) comprised of TFs cascade [40] and used the dataset with 67 time points, including three change points. Note also that the additional 100 datasets were generated by slightly perturbing two original data to avoid the rank deficiency as mentioned above and we used the default values $H = 3$, $W = 0$ and $H = 2$, $W = 0$ for

Table 4.3: Result on inference of change points in *S. cerevisiae* data.

	c_i (Correct answer)	c'_i (DPLSQ-TV)	c'_i (DPLSQ-HS)	ARTIVA
$i = 1$	25	25	25	24
$i = 2$	40	40	40	–
$i = 3$	56	56	56	60
CPU time (sec.)		12359.57	2689.49	–

Table 4.4: Result on inference of change points in *D. melanogaster* data.

	c_i (Correct answer)	c'_i (DPLSQ-TV)	c'_i (DPLSQ-HS)
$i = 1$	31	19	19
$i = 2$	41	31	31
$i = 3$	60	42	42
CPU time (sec.)		59789.33	11039.48

the cell-cycle data and the life-cycle data, respectively.

The performance was evaluated in terms of the time point errors, the modified edges and the average CPU time. With regard to the modified edges, the actual time-varying networks still remain unclear, nevertheless we tried to infer time-varying networks only during the life-cycle of *D. melanogaster* using DPLSQ-HS. In this inference, suppose that the total number of added edges, h approximately equal to the total number of undirected edges predicted by KELLER [40], and thereby applying $h = 100$ and $H = 3$, $W = 0$ for inferring the modified edges of the life-cycle network.

Table 4.3 and Table 4.4 show the results for the time point errors and the CPU time, where c_is are values of change point in the original data and c'_is are estimated values. As can be seen from Table 4.3, there seems to be no difference between the results of DPLSQ-TV and DPLSQ-HS, which can correctly detect the change points where the network topology changes. Moreover, the CPU time required for DPLSQ-HS is about 4 times faster than that needed for DPLSQ-TV. The analysis of *D. melanogaster* shows that both methods can detect the same three change points as listed in Table 4.4. At first glance, readers may think that the errors are large at all change point positions. However, both methods could precisely detect two change points, excluding the case of $i = 3$. From the point of view of computational time, DPLSQ-HS appreciably outperforms DPLSQ-TV; it runs about 5 times faster than DPLSQ-TV. Therefore, DPLSQ-HS allows us to decrease the computation time considerably. These results suggest that, in many cases, DPLSQ-HS can be expected to provide nearly-optimal

Table 4.5: Comparative experiment for the inference of change points.

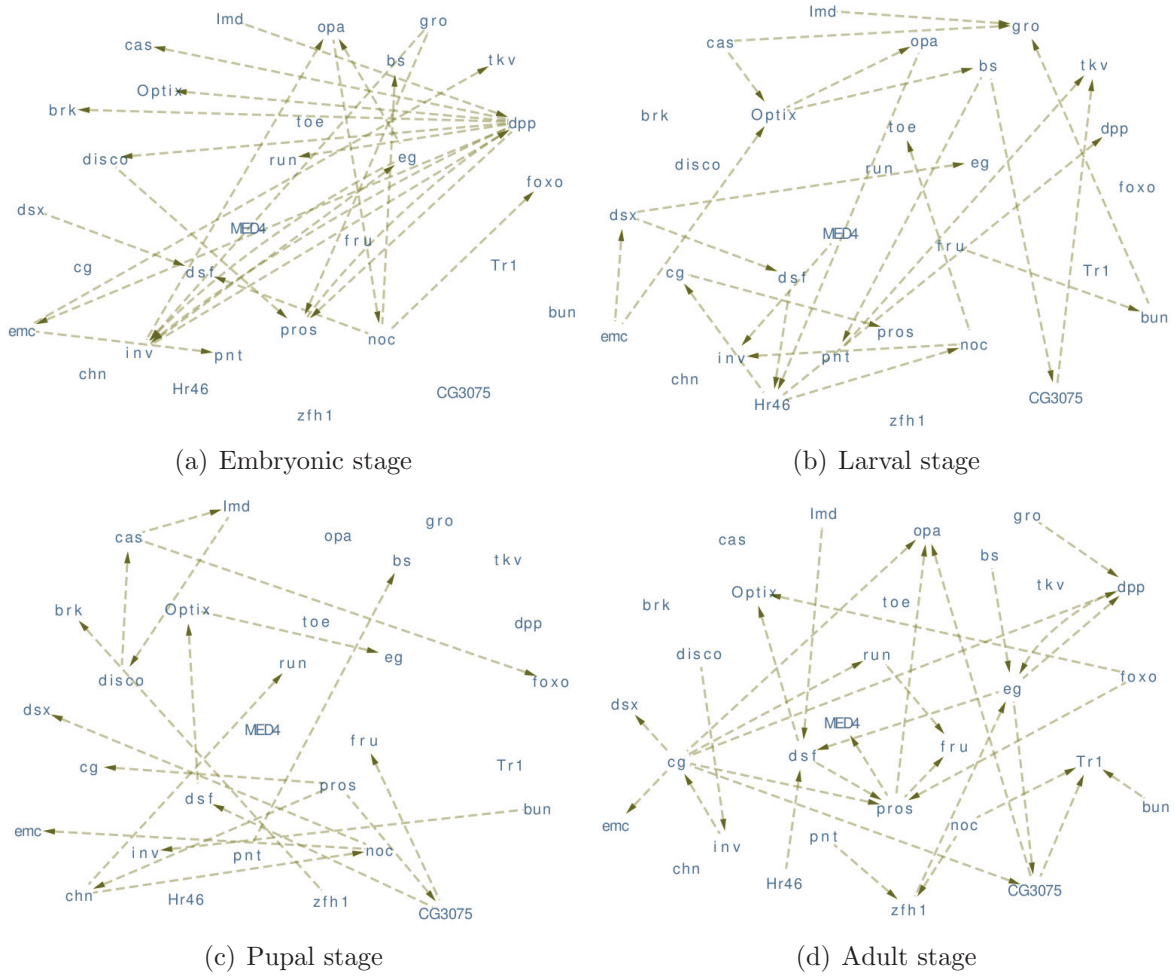
	c_i (Correct answer)	c'_i (DPLSQ-TV)	c'_i (DPLSQ-HS)	ARTIVA
$i = 1$	–	19	19	18 - 19
$i = 2$	31	31	31	31 - 33
$i = 3$	41	42	42	41 - 43
$i = 4$	60	56	56	59 - 61
CPU time (sec.)		1213444.45	26988.79	–

solution at least for change time points and a considerable reduction in computational time.

We also studied the relationship between 30 TFs (Transcriptional Factors) involved in the TFs cascade underlying the nervous system and eye development. The resulting time-varying networks are presented in Figure 4.4. Unfortunately, since the total added edges might be less than the inherent links, we could not identify the obvious feature in time-varying networks during four distinct stages. It should be noted, however, that the gene *pros* appears to be active across the full developmental stages, in addition, two genes *pros* and *dpp* might tend to be highly connected to other genes under the specific stages such as the embryonic and the adult stages. Particulary, in the adult stage, there are several genes that have more than two or three input genes, such as the gene *dsf*, *eg*, *pros* and *dpp*. Overall, DPLSQ-HS was unable to find the function as a hub of gene *Optix* as demonstrated by KELLER. If DPLSQ-HS could be improved the capability to handle large-scale networks which have high indegree, it might reveal more significant relations between the networks.

Next, in comparative analysis with ARTIVA, we employed both the above-mentioned *S. cerevisiae* and *D. melanogaster* microarray datasets, which consist of 71 data points with 10 genes and 67 points with 30 genes, respectively, and tried to identify the change time points. Computational experiments were carried out under the same computational environment as that used in our methods. The results from the yeast microarray data are shown in Table 4.3. Unlike our inference, ARTIVA could detect precisely only two change points, 24 and 60, among three change points, but the second could not.

Lèbrel *et al.* [87] also examined the number of identified change points with *D. melanogaster* data using the ARTIVA algorithm. According to this validation, it has been reported that the time intervals $18 - 19$, $31 - 33$, $41 - 43$ and $59 - 61$ contain more than 40% of all change points. In order to compare with the ARTIVA results, we also attempted to identify four change points using our methods. The results of the comparative experiment using *D. melanogaster* data are summarized in Table 4.5, where c_is are three change points in original data. Although DPLSQ-HS could detect



(a) Embryonic stage

(b) Larval stage

(c) Pupal stage

(d) Adult stage

Figure 4.4: The TFs cascade network consisting of 30 TFs. The dotted arrows represent relationships between two genes. Each (a)-(d) displays the genetic regulation at the corresponding stage and shows remarkably different networks during the distinct stages.

change points similar to those by ARTIVA, the results of ARTIVA appear to have slightly better with respect to the inference of change points than our proposed methods. However, ARTIVA does not detect change time positions but determines time intervals at which the network topology might change. Therefore, DPLSQ-HS is more suited for identifying change time positions at all-time points.

4.4.4 Discussions and Conclusion

We addressed the problem of network completion for time-varying genetic networks from time-series data and proposed two novel methods for solving this problem. Based on the idea of DPLSQ, we developed DPLSQ-TV such that it can perform the modification operations at several time points. In order to detect the change time points and sets of modified edges in network completion, we developed two different types of dou-

ble DP algorithms. The first algorithm DPLSQ-TV, is intended for exact completion and inference of time-varying networks. Although DPLSQ-TV allows us to guarantee the optimality of its solution, it suffers from computational inefficiency as the size of the network grows. In order to improve the computational efficiency of DPLSQ-TV, we developed an effective heuristic method DPLSQ-HS, to reduce the time complexity of calculating least-squared errors by imposing the constraint on the number of combinations of incoming nodes. We showed that each of these two methods works in polynomial time if the maximum indegree is bounded by a constant.

The results of computational experiments reveal that the two methods can detect change points quite precisely and can infer the modified edges with reasonable accuracy. DPLSQ-TV can be expected to provide wide range of applications not only in network completion but also in network inference. Additionally, DPLSQ-HS allowed us to give a relatively good performance in terms of change point detection and to provide a considerable improvement. These results indicate that, in many cases, DPLSQ-HS also enables us to provide near-optimal solutions without increasing the time complexity.

Although DPLSQ-HS is much faster than DPLSQ-TV, it has a drawback: the accuracy was worse than those by DPLSQ-TV, especially, when the observation error level was large and the size of the network increased. Therefore, we need to improve the accuracy of DPLSQ-HS without significantly undermining its efficiency. In our experiments, we specified the number of change time points and the number of edges to be added and deleted. In real use, we may examine several values and select the best one (e.g., the values with the minimum squared errors). However, as discussed in Section 4.4.2, it is likely to lead to overfitting. In order to avoid overfitting, use of AIC (Akaike's Information Criterion) or other information criteria may be useful as discussed in [95] for network inference. Since network completion is more complex than network inference, the method in [95] cannot be directly applied. Therefore, incorporation of an appropriate information criterion into network completion is important future work. Another issue to be tackled is to take into account the relationship between G_i and G_{i+1}. Although G_i and G_{i+1} are inferred independently from the original network G by the proposed methods, there should be some strong relationship between them. Therefore, such an extension is also important future work.

Chapter 5

Network Completion for Static Gene Expression Data

5.1 Background

Estimation of genetic interactions from gene expression microarray data is an interesting and important issue in bioinformatics. There are two types of microarray gene expression data; time-series and non-time-series data as described in Section 2.1.2. A time-series data is a temporal data, which consists of a set of observations measured at successive time instants spaced at uniform time intervals in the same sample and generally exhibits an autocorrelation between successive measurements [48, 49]. It can be used to understand and model the temporal dynamics of behaviors and relationships between genes underlying temporal biological processes such as the life-cycle and the cell-cycle processes. Unfortunately, since the number of observed time points in time-series data is usually too small, various mathematical models and methods to achieve this objective are likely to suffer from low precision. A large amount of temporal data are needed for a reliable estimation. On the other hand, a non time-series data consists of a snapshot of gene expression at a single time point taken from different and independent samples [50]. This type of expression data can usually be obtained through independent tissue samples from healthy individuals and patients of various types with diseases. Although these data are not necessarily static, we may regard them as the static data because these are averaged over a large amount of tissues in rather steady state conditions. For inference of genetic networks, a variety of reverse-engineering methods have been proposed, which are based on different mathematical models as mentioned in Section 1.1. Boolean networks can only be applied to inference of genetic networks from time-series data because the Boolean network is intrinsically a dynamic model. Although Bayesian networks have widely been applied to analysis of static data, they can only output acyclic networks. Many methods have also been proposed using various kinds of differential equation models. However, in many cases, the parameter

estimation will take a large amount of computational time. Overall, most methods suffer from imprecision and/or computational inefficiency and thus there is currently no established or standard method for inference of genetic networks using only gene expression data. Therefore, it is reasonable to try to develop another approach for analysis of gene regulatory networks.

In recent years, there have been several studies and attempts for network completion, not necessarily for biological networks but also for social networks and web graphs. For instance, Kim *et al.* [33] addressed the network completion problem in which an incomplete network including unobserved nodes is given, and then the unobserved parts in a given network should be inferred. They proposed KronEM which combined the Expectation Maximization with the Kronecker graphs model to estimate the missing part of the network. Guimerà *et al.* [35] presented a mathematical method which can identify both missing and spurious interactions in complex networks by using the stochastic block models to capture the structural features in the network. This method was also used to predict yeast protein interaction networks. Hanneke *et al.* [34] defined the network completion as a problem of inferring the rest part of the network, given an observed incomplete network sample and proposed a sampling method to derive confidence intervals from sample networks. In any case, the concept that they have in common is to identify the unobserved or missing part of the network if a certain type of a prototype network is given. As a related work, Saito *et al.* [36] provided a method for measuring the consistency of an inferred network with the observed gene expression data.

Independently, Akutsu *et al.* [37] proposed another model of network completion whose aim is to make the minimum amount of modifications to a given network such that the resulting network is most consistent with the observed data. Based on this concept, we have developed a more practical method DPLSQ, for gene regulatory networks as presented in Chapter 3. DPLSQ is guaranteed to output an optimal solution in polynomial time if the maximum indegree is bounded by a constant. Moreover, we proposed two extension methods named DPLSQ-TV and DPLSQ-HS, to complete and infer time-varying networks as mentioned in Chapter 4. Although these methods are suitable for inferring the dynamics based on the dynamic models, they cannot be applied to network completion or inference from static data.

In this chapter, we propose a novel method DPLSQ-SS (DPLSQ for Static Samples), on the basis of DPLSQ, to complete and infer genetic networks under static conditions using static gene expression data. The purpose of this study is twofold. Firstly, to complete and infer gene regulatory networks under stationary conditions from static expression profile, instead of time-series data and secondly, to investigate the relationship between different kinds of inferred networks under different static conditions (e.g., comparison of normal and cancer networks estimated from samples taken from normal and cancer cells). Static data typically consist of expression levels of various genes,

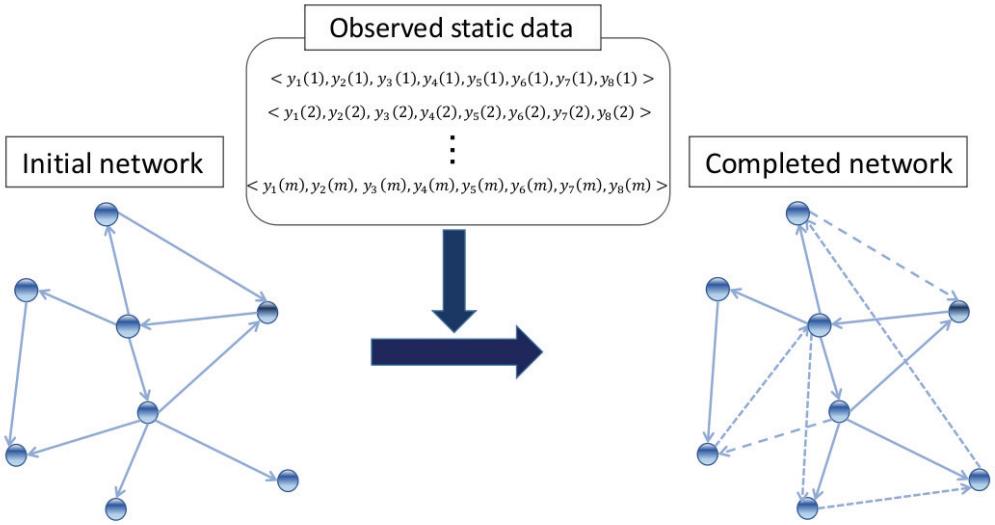


Figure 5.1: Network completion by additions and deletions of edges from m samples. The dotted and the dashed edges represent added and deleted edges, respectively.

which were measured at single time point but for a large amount of different biological samples. As discussed in the beginning part of this section, these types of data can be regarded as the gene expression measurements in stationary phases. Moreover, many of static microarray data are publicly available, in particular, cancer and normal microarray data will be taken from a relatively large size of tumor and normal tissue samples. Therefore, it may be possible to estimate and investigate differences between cancer and normal networks. The basic strategy of DPLSQ-SS is the same as that of DPLSQ: least-squares fitting is used to estimate parameters and dynamic programming is used to minimize the sum-of-squared errors when adding and deleting edges. In order to deal with static data, we modified the error function to be minimized. Although the idea is simple, it brings wider applicability because a large number of static expression data are available. We demonstrate the effectiveness of DPLSQ-SS through computational experiments using synthetic data and microarray data from lung cancer and normal lung tissue samples. We also conduct a performance comparison between DPLSQ-SS and some of state-of-the-art tools using synthetic data.

5.2 Method

In this section, we present a new method DPLSQ-SS, whose aim is to complete and infer genetic networks under stationary conditions from static data. Here we assume additions and deletions of edges as modification operations (see Figure 5.1).

5.2.1 Preliminaries

The purpose of network completion is to modify given networks by adding and deleting edges so as to be consistent with the observed data. In the following, let graph $G(V, E)$ be a given network where V and E are the sets of nodes and directed edges, including loops, respectively. We also let n denote the number of genes and let $V = \{v_i, \cdots, v_n\}$. In this graph G, each node $v_i \in V$ corresponds to a gene and each edge $(v_j, v_i) \in E$ represents a direct regulation between v_i and v_j. For each node v_i, $e^-(v_i)$ and $deg^-(v_i)$, respectively, denote the set of incoming edges to v_i and the number of incoming edges to v_i as defined below:

$$\begin{aligned} e^-(v_i) &= \{v_j | (v_j, v_i) \in E\}, \\ deg^-(v_i) &= |e^-(v_i)|. \end{aligned} \tag{5.1}$$

We also employ the combination of least-squares fitting for estimating parameters and dynamic programming for identifying the network structures. In the following, we will describe the algorithm of the proposed method in details.

5.2.2 Model of Nonlinear Equation and Estimation of Parameters

Since we consider static data and describe the static behavior of genetic interactions, we adopt a mathematical model based on nonlinear equations, instead of differential equations in Chapter 3. We assume that the static state of each gene v_i is determined by the following equation:

$$x_i = a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j} + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j} x_{i_h} + b^i \omega, \tag{5.2}$$

where $v_{i_1}, \cdots, v_{i_d}$ are incoming nodes to v_i, x_i corresponds to the expression value of the i-th gene, and ω denotes a random noise. The second and third terms of the right-hand side of the equation represent linear and nonlinear effects to node v_i, respectively (see Figure 3.2), where the parameters (i.e., a_0^i, a_j^i, $a_{j,h}^i$) can take on any positive and negative values corresponding to an activation effect and an inhibition effect.

We assume that static expression data $\langle y_1(s), y_2(s), \cdots, y_n(s) \rangle$ $(s = 1, \cdots, m)$, are given, where m is the number of samples and $y_i(s)$ denotes the expression value of gene v_i in the s-th sample. The parameters can be estimated by minimizing the following objective function using a standard least-squares fitting method.

$$S^i_{\{v_{i_1}, v_{i_2}, ..., v_{i_d}\}} = \min_{a^i_0, a^i_j, a^i_{j,h}} \sum_{s=1}^{m} \left| y_i(s) - \left(a^i_0 + \sum_{j=1}^{d} a^i_j y_{i_j}(s) + \sum_{1 \le j < h \le d} a^i_{j,h} y_{i_j}(s) y_{j_h}(s) \right) \right|^2 . \tag{5.3}$$

5.2.3 Completion by Addition of Edges

Once the objective function is determined, the completion procedure is the same as that for DPLSQ (see details in Chapter 3). For the simplicity, we start by solving the problem of network completion by adding h edges in total so that the sum-of-squared error is minimized.

We let $\sigma^+_{h_j, j}$ denote the minimum squared error when adding h_j edges to the j-th node and it is defined as

$$\sigma^+_{h_j, j} = \min_{j_1, j_2, ..., j_{h_j}} \left\{ S^j_{e^-(v_j) \cup \{v_{j_1}, v_{j_2}, ..., v_{j_{h_j}}\}} \right\}, \tag{5.4}$$

where each v_{j_l} must be selected from $V - v_j - e^-(v_j)$. In order to avoid combinatorial explosion, we constrain the maximum h_j is a small constant H and let $\sigma^+_{h_j, j} = +\infty$ for $h_j > H$ or $h_j + deg^-(v_j) \ge n$.

Here, we define $D^+[h, i]$ by

$$D^+[h, i] = \min_{h_1 + h_2 + \cdots + h_i = h} \sum_{j=1}^{i} \sigma^+_{h_j, j}. \tag{5.5}$$

The elements of array $D^+[h, j]$ can be computed using dynamic programming as follows:

$$\begin{aligned} D^+[h, 1] &= \sigma^+_{h,1}, \\ D^+[h, j+1] &= \min_{h' + h'' = h} \{ D^+[h', j] + \sigma^+_{h'', j+1} \}. \end{aligned} \tag{5.6}$$

It must be noted that $D^+[h, n]$ is determined uniquely regardless of the ordering of nodes in the network. The correctness of this algorithm based on dynamic programming can be proven by Equation (5.7),

$$\begin{aligned} \min_{h_1 + h_2 + \cdots + h_n = h} \sum_{j=1}^{n} \sigma^+_{h_j, j} &= \min_{h' + h'' = h} \left\{ \min_{h_1 + h_2 + \cdots + h_{n-1} = h'} \sum_{j=1}^{n-1} \sigma^+_{h_j, j} + \sigma^+_{h'', n} \right\} \\ &= \min_{h' + h'' = h} D^+[h', n-1] + \sigma^+_{h'', n}. \end{aligned} \tag{5.7}$$

5.2.4 Completion by Addition and Deletion of Edges

The above mentioned algorithm can be extended for the network completion problem by adding and deleting of edges.

We let $\sigma_{h_j, w_j, j}$ denote the minimum squared error when adding h_j edges to $e^-(v_j)$ and deleting w_j edges from $e^-(v_j)$ as below, where added and deleted edges must be disjoint.

$$\sigma_{h_j, w_j, j} = \min_{\substack{j_1, j_2, ..., j_{h_j} \\ j'_1, j'_2, ..., j'_{w_j}}} \left\{ S^j_{e^-(v_j) - \{v_{j'_1}, v_{j'_2}, ..., v_{j'_{w_j}}\} \cup \{v_{j_1}, v_{j_2}, ..., v_{j_{h_j}}\}} \right\}, \tag{5.8}$$

where $\{v_{j'_1}, v_{j'_2}, \ldots, v_{j'_{w_j}}\}$ is the set of deleted edges from $e^-(v_j)$. As described in Section 5.2.3, we also impose constraints (i.e., the maximum h_j and w_j are small constants H and W). We let $\sigma_{h_j, w_j, j} = +\infty$ if $h_j > H$, $w_j > W$, $h_j - w_j + deg^-(v_j) \geq n$, or $h_j - w_j + deg^-(v_j) < 0$ holds. Therefore, the problem is stated as term (5.9),

$$\min_{\substack{h_1 + h_2 + \cdots + h_n = h \\ w_1 + w_2 + \cdots + w_n = w}} \sum_{j=1}^{n} \sigma_{h_j, w_j, j}. \tag{5.9}$$

Here, we define $D[h, w, i]$ by

$$D[h, w, i] = \min_{\substack{h_1 + h_2 + \cdots + h_i = h \\ w_1 + w_2 + \cdots + w_i = w}} \sum_{j=1}^{i} \sigma_{h_j, w_j, j}. \tag{5.10}$$

Then, the network completion problem can be solved by using the dynamic programming algorithm as follows:

$$\begin{aligned} D[h, w, 1] &= \sigma_{h, w, 1}, \\ D[h, w, j+1] &= \min_{\substack{h' + h'' = h \\ w' + w'' = w}} \{D[h', w', j] + \sigma_{h'', w'', j+1}\}. \end{aligned} \tag{5.11}$$

5.2.5 Time Complexity Analysis

We will also discuss the computational complexity of DPLSQ-SS for network completion by additions and deletions of edges.

The algorithm for least-squares is known to be done with time complexity $O(mp^2 + p^3)$ where m and p are the number of samples and parameters (refer to Section 2.2.4). In our proposed method, since we assume that the maximum indegree in a given network and the number of parameters are bounded by constants. In this case, the time complexity per least-squares fitting can be estimated as $O(m)$.

Next we analyze the time complexity required for $\sigma_{h_j, w_j, j}$ and $D[h, w, i]$. The time complexity required for computation of $\sigma_{h_j, w_j, j}$ is $O(mn^{H+1})$ (see details in Section

3.2.6), where the time complexity of computing the minimum least-squared error for j-th node depends on the upper bounds for the number of adding and deleting edges per node, H and W. In addition, the time complexity for $D[h, w, i]$s is $O(n^3)$, when considering that the size of table $D[h, w, i]$ is $O(n^3)$. Therefore, the overall time complexity for DPLSQ-SS is

$$O(mn^{H+1} + n^3). \tag{5.12}$$

This analysis suggests that DPLSQ-SS can be suitable for genetic networks if $H \leq 2$ and n is not too large.

If the maximum indegree of the initial network is not bounded by a constant, the time complexity per least-squares fitting increases to $O(mn^4 + n^6)$ and the number of combinations to be examined per node increases to $O(n^{H+W})$, as mentioned in Section 3.2.6. In this case, the total time complexity would be $O(n^{H+W+1} \cdot (mn^4 + n^6))$, which suggests that network completion should not start with dense networks but with sparse networks.

5.3 Results and Discussion

5.3.1 Availability

To evaluate the effectiveness of DPLSQ-SS, we performed two types of computational experiments using both synthetic data and microarray expression data. All experiments were carried out on a PC with Intel Core (TM)2 Quad CPU Q9650 (3.0 GHz) with 186 GB RAM running under the Linux Kernel. We employed the liblsq library (http://www2.nict.go.jp/aeri/sts/stmg/K5/VSSP/ install_lsq.html) for a least-squares fitting method.

5.3.2 Inference Using Synthetic Data

In order to assess the potential effectiveness of DPLSQ-SS, we begin with network inference using two kinds of synthetic data. Recall that network completion beginning with a null network corresponds to network inference.

We employed here nonlinear equations as gene regulation rules between genes. Since it is hard to generate static data by numerical simulations, we created manually nonlinear equations with obvious solutions as the synthetic network topology and regarded each solution as static data for one sample. In other words, if we make n equations with n variables, it is assumed that there exist n genes in the synthetic network. We give an example of nonlinear equations with 3 variables below:

$$\begin{aligned} x_1 &= x_1^2 - 6, \\ x_2 &= x_2^2 - 2, \\ x_3 &= x_1 x_2 + 1, \end{aligned} \tag{5.13}$$

where we assume that $x_i (i = 1, \cdots, 3)$ corresponds to the expression value of i-th gene. Therefore, an example network consists of 3 genes with 4 edges, including self-loops. We can find four solutions below by solving this set of equations and will employ these solutions as synthetic static data.

$$(3, 2, 7), (3, -1, -2), (-2, 2, -3), (-2, -1, 3). \tag{5.14}$$

Also note that since the use of static data consisting only of a few solutions easily cause rank deficiency, we generated additional 400 data sets for each solution by adding random numbers uniformly distributed between -0.5 and 0.5.

Under the above model, we examined DPLSQ-SS for network inference, using synthetic data which is generated as described above and letting $E = \emptyset$ in the initial network. It should be noted that we let upper bounds for the number of adding and deleting edges per node, $H = 2$ and $W = 0$, respectively. Furthermore, in order to examine how the CPU time changes as the size of the network grows, we generated several synthetic networks with 10 and 20 nodes as original networks by making the nonlinear equation with corresponding number of variables.

Since the number of added edges was always equal to the number of edges in the original network, we conducted the performance evaluation of DPLSQ-SS by means of the averaged accuracy, which was defined as the ratio of correctly inferred edges to total edges in the original network (i.e., the number of added edges) and the averaged computational time over 5 modified networks.

We also compared DPLSQ-SS against two existing tools for inference of genetic networks, ARACNE [27, 28] and GeneNet [70, 71]. The details about two existing methods were described previously in Section 3.3.3. We employed datasets which were generated by the same way for DPLSQ-SS and default parameter settings for both tools. We evaluated the results by the ratio of correctly inferred edges and the averaged CPU time (see Table 5.1). The CPU time used by ARACNE is user time + sys time and that used by GeneNet is the difference in system time between start and end.

The results on DPLSQ-SS and comparative methods using synthetic data show that the accuracies by DPLSQ-SS are higher than those by ARACNE and GeneNet. Although ARACNE cannot handle networks with self-loops but GeneNet can, both methods showed almost the same performance in the case of $n = 10$. On the whole, three methods have something in common, which perform with low accuracy as the size of the network grows. As for the CPU time, ARACNE was slightly faster than DPLSQ-SS and

Table 5.1: Comparison of our method with other existing methods.

		Method		
		DPLSQ-SS	ARACNE	GeneNet
$n = 10$	Accuracy	0.779	0.578	0.571
	CPU time (sec.)	1.784	1.113	4.020
$n = 20$	Accuracy	0.722	0.554	0.390
	CPU time (sec.)	14.482	4.795	4.040

GeneNet in case of $n = 10$. In addition, the CPU time by DPLSQ-SS increases rapidly as the size of the network grows, in contrast to those by the comparative methods. Since DPLSQ-SS works in polynomial time, if we obtain sufficient computer resource, DPLSQ-SS can handle large-scale networks. Since the accuracy is the most important criterion and DPLSQ-SS is more accurate than existing methods, our proposed method might be a useful tool for network inference.

5.3.3 Inference Using DREAM4 Data

In this subsection, we perform a comparison with other methods in order to perform an "unbiased" evaluation since the results in Section 5.3.2 are based only on the nonlinear equation model allowing self-loops adopted by DPLSQ-SS. In the actual genetic interactions, various-types of feedback and feedforward regulations exist for characterizing the dynamics of biological systems. In this experiment, we used synthetic datasets generated by GeneNetWeaver (GNW) [96], which provide benchmarks and performance testing for network inference methods in the DREAM (Dialogue on Reverse Engineering Assessment and Methods) challenge (http://www.the-dream-project.org/challenges). One aim of the DREAM project is to provide benchmark data on real and simulated expression data for network inference. This project organizes several editions, where GNW has been developed to generate simulated genetic network motifs and simulated expression data. In this evaluation, we used the DREAM4 challenge which is divided into three subchallenges called InSilico_Size10, InSilico_Size100, and InSilico_Size100_Multifactorial, consisting of five networks.

We validated the performance using InSilico_Size10 subchallenge consisting of gold standard 10 gene networks and simulated expression data generated under different conditions (wild-type, knockouts, knockdowns, multifactorial perturbations, and time-series). Since only one set of wild-type of data, which corresponds to static data, is provided for each network and it is not enough to obtain a reliable inference, we generated additional 500 static data sets in the same way as described in Section 5.3.2. Table

Table 5.2: Results on network inference with DREAM4 data, where the accuracy is shown for each case.

	Method		
	DPLSQ-SS	ARACNE	GeneNet
Insilico_size_10_1	0.2666	0.2000	0.0666
Insilico_size_10_2	0.1875	0.2500	0.1250
Insilico_size_10_3	0.1333	0.2000	0.0666
Insilico_size_10_4	0.1538	0.3076	0.0769
Insilico_size_10_5	0.0833	0.1666	0.0833

5.2 summarizes the results of three methods in terms of the accuracy defined in Section 5.3.2. As can be seen from this table, the performance of any method is not good. It is reasonable because inference was preformed based on one set of expression data (i.e., $m = 1$) even though perturbed data were also used. Although ARACNE was better than DPLSQ-SS in four cases, DPLSQ-SS was better than ARACNE in one case, in addition, our method outperformed GeneNet in many cases. This result suggests that although DPLSQ-SS is not necessarily the best for simulated data in DREAM4, it has reasonable performance when a very few samples are given. If we demonstrate the effect of combining the wild-type data with the knockout data for each network, we may get some interesting results.

5.3.4 Completion Using Synthetic Data

We also examined network completion using synthetic data. In this experiment, we adopted the nonlinear equations described in Section 5.3.2. In order to examine network completion, we applied synthetic networks to DPLSQ-SS, which are generated by randomly adding h edges and deleting w edges in total from an original network. The performance is evaluated in terms of the accuracy of modified edges and the computational time for network completion. The accuracy is defined as follows:

$$\frac{h + w + |E_{orig} \cap E_{cmpl}| - |E_{orig}|}{h + w}, \tag{5.15}$$

where E_{orig} and E_{cmpl} are the set of edges in the original network and the completed network, respectively. This value takes 1 or 0 if all edges are correctly or incorrectly added and deleted. For each (h, w), we took the averaged accuracy and the CPU time for completion over 5 modifications for 10 and 20 networks, where we used the default values of $H = W = 2$. Note that we also generated additional 400 data sets for each

Table 5.3: Results on completion with synthetic data on DPLSQ-SS.

	No. added edges, No. deleted edges		
		Accuracy	CPU time (sec.)
$n = 10$	$h = 1,\ w = 1$	1.000	0.720
	$h = 2,\ w = 2$	1.000	5.810
	$h = 3,\ w = 3$	1.000	4.610
	$h = 4,\ w = 4$	1.000	5.000
	$h = 5,\ w = 5$	0.700	4.410
$n = 20$	$h = 1,\ w = 1$	1.000	2.760
	$h = 2,\ w = 2$	1.000	51.870
	$h = 3,\ w = 3$	0.833	46.220
	$h = 4,\ w = 4$	1.000	53.880
	$h = 5,\ w = 5$	0.700	48.910

static solutions by adding random numbers uniformly distributed between -0.5 and 0.5 to avoid rank deficiency.

As can be seen from the Table 5.3 that DPLSQ-SS has quite high accuracy regardless of the number of h and w except for $h = w = 5$. Moreover, the CPU time increases rapidly when applied to networks with 20 genes. In comparison with the CPU time for network inference by DPLSQ-SS, there seems to be a remarkable difference even if n equals 10. Obviously, the number of modified edges for network inference is much larger than that for network completion. However, network completion requires a lot of CPU time than network inference (refer to the Section 5.3.2). In this experiment, we used the default values of $H = 2$, $W = 2$ for network completion and $H = 2$, $W = 0$ for network inference. This result suggests that the time complexity of DPLSQ-SS depends not so much on the number of modified edges, h and w, but depends much on the number of H and W as indicated in Section 5.2.5.

5.3.5 Inference Using Real Data

We also examined DPLSQ-SS for inference of genetic networks from static microarray data under multiple conditions. The aim of this experiment is to complete and infer different static gene networks under different conditions and investigate the differences of these network topologies. We focus on the genetic network related to lung cancer, because most available data on cancer or multiple disease is static [97] and employed a partial network as a gold standard, which contains RB/E2F pathway in human small cell lung carcinoma (SCLC) from the KEGG pathway database [51] shown in Figure

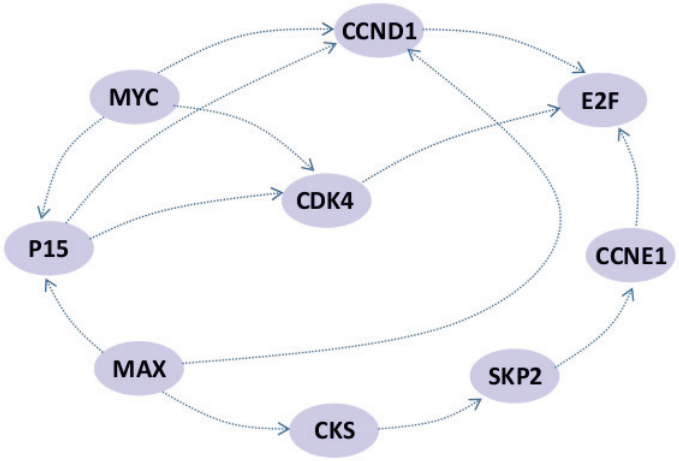


Figure 5.2: A part of small cell lung cancer network, containing RB/E2F pathway.

5.2. The RB/E2F pathway is one of two major tumor suppressor pathways and the retinoblastoma (RB) gene plays an important role in cancer development [98]. The RB gene is known to control the activity of E2F transcription factor which is implicated in regulating cell-cycle progression under the control of cyclin-CDK complexes such as CDK4, CCND1 and CCNE1 (see Figure 5.2). It is also clear that the E2F family plays a crucial role and determines the rate of proliferation in both normal and tumor cells [99, 100] and that the loss of E2F leads to cancer formation. In this way, it is obvious that the gene abnormality of RB/E2F pathway is linked to cancer risk, and this suggests that there must be distinct differences between the cancer and the normal genetic networks.

In this experiment, we examine whether DPLSQ-SS can distinguish the difference between the cancer and the normal networks by using static expression data. We considered the network consisting 9 genes as the original network, which were selected from KEGG database and Entrez Gene ID and we summarized the gene symbols and notations which are referred by RefGene (http://refgene.com/) as shown in Table 5.4. As for the static data, we used static microarray data for lung cancer tissue samples obtained by Beer *et al.* [101]. They performed hierarchical clustering to analyze gene expression profiles from lung adenocarcinoma tumor tissues and those from normal lung tissues. This dataset comprises 86 tumor and 10 normal samples and is publicly available from the study of Choi *et al.* [102]. In order to examine the relationship between cancer and normal network topologies, we performed network inference using these two types of data, where $H = 2$, $W = 0$, $h = 13$ and $w = 0$. In order to avoid rank deficiency, we also created additional 5 data sets for each expression value by adding random numbers uniformly distributed between -0.5 and 0.5. The results are

Table 5.4: Summary of gene symbols and notations. The RB gene did not listed in this table, because there is no expression value of the RB gene in the raw data file.

Number	Gene symbol	Gene annotation
1	MYC	v-myc myelocytomatosis viral oncogene homolog (avian)
2	P15	cyclin-dependent kinase inhibitor 2B
3	CDK4/6	cyclin-dependent kinase 4
4	CCND1	cyclin D1
5	MAX	MYC associated factor X
6	CKS1	CDC28 protein kinase regulatory subunit 1B
7	SKP2	S-phase kinase-associated protein 2 (p45)
8	CCNE1	cyclin E1
9	E2F3	E2F transcription factor 3

displayed in Figure 5.3. We also compared DPLSQ-SS with ARACNE and GeneNet using these microarray data and regarded the network shown in Figure 5.2 as the correct network. The result is shown in Table 5.5, where the accuracy (i.e., the ratio of the number of correctly inferred edges to the number of added edges) was defined in Section 5.3.2.

From the Table 5.5, DPLSQ-SS was worse than ARACNE for the cancer network, but it was better for the normal network. On the contrast, DPLSQ-SS works better than GeneNet for both networks. As can be seen from Figure 5.3, while DPLSQ-SS has some difference in the accuracy between the cancer and the normal networks, existing methods show the similar accuracies, of course, these accuracies were estimated on the basis of the unified criterion (i.e., SCLC network provided by the KEGG database shown in Figure 5.2). Considering the crucial link between genes and cancer, there probably are significant differences between the cancer and the normal networks. Therefore, these results suggest that DPLSQ-SS provides reasonable estimates on microarray expression data compared with existing methods.

(b) Cancer network

(a) Normal network

Figure 5.3: Results on inference with gene expression data for cancer and normal samples. We could verify that the E2F activity is normally regulated by the RB/E2F pathway activities in the normal network and does not seem to be regulated by the RB/E2F pathway in the cancer network. The red bold solid arrows represent the correctly added edges. The blue bold dashed arrows denote the incorrectly added edges (i.e., added edges that are included in the original network).

Table 5.5: Results on inference of real networks, where the accuracy is shown for each case. Since directions of edges are ignored in comparison with ARACNE and GeneNet, bold dashed arrows (refer to Figure. 5.3) connecting to P15 are regarded as correct.

	Method		
	DPLSQ-SS	ARACNE	GeneNet
Normal network	0.3846	0.3076	0.0769
Cancer network	0.1538	0.3846	0.0769

The inferred normal network indicated the presence of RB/E2F pathway involved in the regulation of E2F activity. We also found that the tumor suppressor gene P15 controlled the CDK4 activity and E2F was under the regulation of both CDK4 and CCND1. On the other hand, in the inferred cancer network, no significant correlation was observed between genes in the RB/E2F pathway. Instead, we discovered the regulation of CCND1 and disruption of the RB/E2F pathway. It has been reported that overexpression of CDK4/6 and CCND1 and disruption of the RB/E2F pathway could contribute to cancer progression [98, 103]. Therefore, inference of two types of networks could make the reasonable estimate that nearly matches biological knowledge as mentioned above and could capture the features of each network. Although there is no common edge between the inferred cancer network and the original network, it is reasonable because cancer networks may be very different from normal networks. This result suggests that our proposed method can infer different static networks under the different conditions and can identify the structural features of cancer and normal networks.

5.3.6 Discussions and Conclusion

In this chapter, we addressed the problem of completing and inferring genetic networks under stationary conditions from static gene expression data. We defined this problem as modifying a given network so that the inferred network matches the observed data. The goal of this study is (1) to complete static network using static data and (2) to examine the differences between two types of genetic networks under different conditions. In order to achieve our goal, we proposed a novel method called DPLSQ-SS, based on least-squares fitting and dynamic programming. This method works in polynomial time if the maximum indegree is bounded by a constant. We demonstrated the effectiveness of DPLSQ-SS through computational experiments using synthetic data and microarray data. In particular, we tried to infer and compare the cancer and the normal networks from static expression data. As the results using synthetic data, DPLSQ-SS showed relatively good performance in comparison to other existing methods. As the

results using microarray data obtained from normal and cancer samples, it is seen that DPLSQ-SS allows us to distinguish the differences between genetic networks under different conditions.

There is some room for extending DPLSQ-SS. For example, we employed here simple nonlinear equations as the model of gene regulation rules, but it can be replaced by equivalent complicated equations. Although DPLSQ-SS works in polynomial time, the degree of polynomial is not low, which prevents the method from being applied to completion of large-scale networks. However, DPLSQ-SS can be highly parallelizable: $\sigma_{h_j,w_j,j}$ can be computed independently for each $\sigma_{h_j,w_j,j}$. Therefore, parallel implementation makes DPLSQ-SS more powerful and is also important future work. Although we have focused on completion and inference of gene regulatory networks, completion and inference of large-scale protein-protein or ChIP-chip/seq interaction networks are also important. If we would apply DPLSQ-SS to various types of interactions in biological networks other than those of genes, it will be more suitable for use in practice. Since the proposed method is currently only applicable to gene regulatory networks, extension and application of DPLSQ-SS for these networks should be studied in the future work.

Chapter 6

Conclusion

Gene expression is controlled by a complex set of interactions between cellular components such as genes, proteins, and RNA molecules. Suppose that these molecules would regulate one another like a form of "regulatory network", the elucidation of gene regulatory network helps us to gain a better understanding of the dynamic/static behavior of genetic systems and their mechanisms in gene expression. General approaches for genetic network inference mostly aim to reconstruct gene regulatory networks comprising multiple interactions among genes using mathematical models based on expression information derived from microarray experiments. This type of approach can build networks only from microarray observations and can indeed exhibit specific aspects of gene expression, however, in many cases, these predictions appear to be vague or unrealistic for faithfully describing the actual gene expression. Needless to say, a more reasonable and realistic approach should be developed. In this thesis, we introduced a novel approach for reverse-engineering of gene regulatory networks, network completion, and proposed three novel computational methods for solving different kinds of network completion problems.

The goal of network completion is to make the minimum modifications (additions and deletions of edges) to a well-known given network such that the resulting network is most consistent with the observations, according to the basic concept in the study [37]. In Chapter 3, we considered a problem of network completion from time-series data and proposed a new method DPLSQ, based on the differential equation model. The application of least-squares fitting enables us to reduce network completion to an optimization problem whose objective is to minimize the sum-of-squared errors between the predicted and the observed values when adding and deleting edges. Moreover, by applying dynamic programming, DPLSQ can automatically determine appropriate modifications (find an optimal solution) if the total numbers of adding and deleting edges are given. The experimental analysis showed that DPLSQ provided good performance in terms of the accuracy of modified edges and found exact solutions in polynomial time subject to the constraints. It should be noted that the optimality

of its solution is guaranteed in DPLSQ under the assumption that the expression patterns of all nodes are observable while existing methods do not guarantee. Therefore, we conclude that DPLSQ can provide an optimal solution within a reasonable time and will be suitable for practical applications by considering the extensions for large-scale networks.

In Chapter 4, we presented a network completion problem for time-varying genetic networks from time-series data and proposed two methods DPLSQ-TV and DPLSQ-HS, by extending DPLSQ. DPLSQ-TV was an exact method for detecting change points where topological changes may occur and modified edges, adopting a novel double dynamic programming algorithm so as to make modifications at several time points. DPLSQ-TV guaranteed to the optimality of its solution in polynomial time, whereas it suffered from low computational efficiency, because of the large number of calculations of sum-of-squared errors. To overcome this weakness, we also proposed a heuristic method DPLSQ-HS, to reduce the computational time by imposing the constraint on the number of combinations of incoming edges.

The results of completion with synthetic data, DPLSQ-TV showed high precision for detecting change points and relatively good precision for modifications of edges and DPLSQ-HS also provided near-optimal performance with reduced time complexity. With regard to microarray analysis, both methods were able to consistently identify same change points and outperform the existing method ARTIVA. Therefore, DPLSQ-HS can be expected to give a reasonably good approximation for the temporal behavior analysis of gene regulatory networks.

It must be noted that the time complexity of DPLSQ-TV does not have low-degree polynomials, thereby preventing it from applying large-scale networks. DPLSQ-TV should also be capable of parallel processing, that is the minimum sum-of-squared error for each node can be calculated independently. Therefore, extensions for parallel implementation are needed for practical applications of inference of large-scale genetic networks.

In Chapter 5, we considered a new problem of completing and inferring genetic networks under stationary conditions from static expression profiles and proposed a novel method DPLSQ-SS, according to the main idea of DPLSQ. In order to cope with the static data, we adopted the nonlinear equation model to express the static behaviors on genetic regulations. This problem can also be solvable in polynomial time using least-squares fitting in combination with dynamic programming.

The experimental results on synthetic data indicated that DPLSQ-SS outperformed existing methods in reasonable CPU times for network inference. Moreover, it provided more accurate estimates on completing genetic networks from static data with polynomial time complexity depending on the maximum number of adding edges per node. In addition, in order to examine the relationship between two types of static networks, we also verified whether DPLSQ-SS could distinguish the topological differences between

the cancer and the normal networks under static conditions from cancer and normal expression data. The results revealed that DPLSQ-SS could capture distinctive features of different types of static networks, such as the cancer and the normal networks.

Note that parallel implementation of DPLSQ-SS is also effective for developing large-scale networks. Static microarray data typically measures just once expression values of population individuals, such as normal people/tissues and patients/tissues of variety type of disease, and most of them are publicly available. Understandably, if a well-known network is given as an initial network, DPLSQ-SS must be able to complete the normal network (well-known network) based on information about expression patterns derived from disease samples. Therefore, inference of static networks under different conditions will most likely help us to gain a better understanding of the relationship between disease progression and alteration of gene expression.

Finally, we would like to emphasize that network completion is an innovative and useful approach to describe more naturally gene regulatory systems closer to actual phenomena based on the expression data. Since reverse-engineering has desirable properties to reconstruct the connectivities between genes from experimental observations, the valuable information extracted from observed data should be fully taken advantage of. Furthermore, another important aspect of this approach is that even if microarray data may contain measurements of unknown genes, network completion methods might be able to predict functions of un-characterized genes. Therefore, network completion approach is expected to open up new possibilities for understanding of complex phenomena which arise in real genetic interactions and new insights will also be expected to make contributions to the fields of biology and medicine.

Bibliography

[1] F. Jacob, and J. Monod, "Genetic regulatory mechanisms in the synthesis of proteins," *Journal of Molecular Biology*, 3(3):318-356, 1961.

[2] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed nets," *Journal of Theoretical Biology*, 22(3):437-467, 1969.

[3] J. S. Lee, and B. Kahng, "Cell-cycle checkpoints in a state network," *Journal of Korean Physical Society*, 52:S193-S196, 2008.

[4] W. P. Lee, and W. S. Tzou, "Computational methods for discovering gene networks from expression data," *Briefings In Bioinformatics*, 10(4):408-423, 2009.

[5] T. Akutsu, S. Miyano, and S. Kuhara, "Inferring qualitative relations in genetic networks and metabolic pathways," *Bioinformatics*, 16(8):727-734, 2000.

[6] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, 18(2):261-274, 2002.

[7] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, 7(3-4):601-620, 2000.

[8] S. Imoto, T. Goto, and S. Miyano, "Estimation of genetic networks and functional structures between genes by using Bayesian network and nonparametric regression," *Proceedings of the Pacific Symposium on Biocomputing*, 7:175-186, 2002.

[9] S. Imoto, S. Kim, T. Goto, S. Aburatani, K. Tashiro, S. Kuhara, and S. Miyano, "Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network," *Journal of Bioinformatics and Computational Biology*, 1(2):231-252, 2003.

[10] R. Neapolitan, "Learning Bayesian Networks," *Prentice-Hall, Inc.*, Upper Saddle River, NJ, USA, 2003.

[11] R. Manshaei, P. S. Bidari, M. A. Shoorehdeli, A. Feizi, T. Lohrasebi, M. A. Malboobi, M. Kyan, and J. Alirezaie, "Hybrid-controlled neurofuzzy networks analysis

resulting in genetic regulatory networks reconstruction," *International Scholarly Research Network Bioinformatics*, 2012:419419, 2012.

[12] N. Dojer, A. Gambin, A. Mizera, B. Wilczyński, and J. Tiuryn, "Applying dynamic Bayesian networks to perturbed gene expression data," *BMC Bioinformatics*, 7:429, 2006.

[13] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young, "Bayesian methods for elucidating genetic regulatory networks," *IEEE Intelligent Systems*, 17(2):37-43, 2002.

[14] S. Y. Kim, S. Imoto, and S. Miyano, "Inferring gene networks from time series microarray data using dynamic Bayesian networks," *Briefings in Bioinformatics*, 4(3):228-235, 2003.

[15] M. Zou, and S. D. Conzen, "A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data," *Bioinformatics*, 21(1):71-79, 2005.

[16] C. Yuan, and B. Malone, "An improved admissible heuristic for learning optimal Bayesian networks," *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2012:924-933, 2012.

[17] I. A. Maraziotis, A. Dragomir, and D. Thanos, "Gene regulatory networks modeling using a dynamic evolutionary hybrid," *BMC Bioinformatics*, 11:140, 2010.

[18] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics*, 16(8):707-726, 2000.

[19] Y. Wang, T. Joshi, X. S. Zhang, D. Xu, and L. Chen, "Inferring gene regulatory networks from multiple microarray datasets," *Bioinformatics*, 22(19):2413-2420, 2006.

[20] M. Bansal, G. D. Gatta, and D. Bernardo, "Inference of gene regulatory networks and compound mode of action from time course gene expression profiles," *Bioinformatics*, 22(7):815-822, 2006.

[21] D. di. Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. Eastwood, A. P. Wojtovich, S. J. Elliott, S. E. Schaus, and J. J. Collins, "Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks," *Nature Biotechnology*, 23(3):377-383, 2005.

[22] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, and A. Konagaya, "Inference of S-system

models of genetic networks using a cooperative coevolutionary algorithm," *Bioinformatics*, 21(7):1154-1163, 2005.

[23] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita, "Dynamic modeling of genetic networks using genetic algorithm and S-system," *Bioinformatics*, 19(5):643-650, 2003.

[24] A. Zaslaver, A. E. Mayo, R. Rosenberg, P. Bashkin, H. Sberro, M. Tsalyuk, M. G. Surette, and U. Alon, "Just-in-time transcription program in metabolic pathways," *Nature Genetics*, 36(5):486-491, 2004.

[25] S. Wu, Z. P. Liu, X. Qiu, and H. Wu, "Modeling genome-wide dynamic regulatory network in mouse lungs with influenza infection using high-dimensional ordinary differential equations," *PLoS One*, 9(5):e95276, 2014.

[26] H. Chen, D. A. K. Maduranga, P. A. Mundra, and J. Zheng, "Integrating epigenetic prior in dynamic Bayesian network for gene regulatory network inference," *Proceedings of Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 76-82, 2013.

[27] A. A. Margolin, K. Wang, W. K. Lim, M. Kustagi, I. Nemenman, and A. Califano, "Reverse engineering cellular networks," *Nature Protocols*, 1(2):662-671, 2006.

[28] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano, "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, 7(1):S7, 2006.

[29] M. Hecker, S. Lambeck, S. Toepfer, E. van Someren, and R. Guthke, "Gene regulatory network inference: data integration in dynamic models - a review," *BioSystems*, 96(1):86-103, 2009.

[30] S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano, "Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks," *Journal of Bioinformatics and Computational Biology*, 2(1):77-98, 2004.

[31] A. Bernardo, and A. J. Hartemink, "Informative structure priors: joint learning of dynamic regulatory networks from multiple types of data," *Proceedings of the Pacific Symposium on Biocomputing*, 459-470, 2005.

[32] A. V. Werhli, and D. Husmeier, "Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge," *Statistical Applications in Genetics and Molecular Biology*, 6(1):15, 2007.

[33] M. Kim, and J. Leskovec, "The network completion problem: inferring missing nodes and edges in networks," *Proceedings of 2011 SIAM International Conference on Data Mining*, 47-58, 2011.

[34] S. Hanneke, and E. P. Xing, "Network completion and survey sampling," *Journal of Machine Learning Research*, 5:209-215, 2009.

[35] R. Guimerà, and M. S. Pardo, "Missing and spurious interactions and the reconstruction of complex networks," *Proceedings of the National Academy of Sciences of the United States of America*, 106(52):22073-22078, 2009.

[36] S. Saito, S. Aburatani, and K. Horimoto, "Network evaluation from the consistency of the graph structure with the measured data," *BMC Systems Biology*, 2(84), 2008.

[37] T. Akutsu, T. Tamura, and K. Horimoto, "Completing networks using observed data," *Proceedings of 20th International Conference on Algorithmic Learning Theory*, 126-140, 2009.

[38] Oxford Dictionaries, http://www.oxforddictionaries.com/definition/english/complete.

[39] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. R. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young, "Transcriptional regulatory networks in Saccharomyces cerevisiae," *Science*, 298:799-804, 2002.

[40] L. Song, M. Kolar, and E. P. Xing, "KELLER: estimating time-varying interactions between genes," *Bioinformatics*, 25(12):i128-i136, 2009.

[41] P. Huijser, and M. Schmid, "The control of developmental phase transitions in plants," *Development*, 138(19):4117-4129, 2011.

[42] S. Knudsen, "Cancer diagnostics with DNA microarrays," *John Wiley and Sons, Inc.*, Hoboken, NJ, USA, 2006.

[43] S. Knudsen, "Guide to analysis of DNA microarray data," *John Wiley and Sons, Inc.*, Hoboken, NJ, USA, 2004.

[44] A. L. Tarca, R. Romero, and S. Draghici, "Analysis of microarray experiments of gene expression profiling," *American Journal of Obstetrics and Gynecology*, 195(2):373-388, 2006.

[45] V. Trevino, F. Falciani, and H. A. Barrera-Saldaña, "DNA microarrays: a powerful genomic tool for biomedical and clinical research," *Molecular Medicine*, 13(9-10):527-541, 2007.

[46] D. Schottenfeld, and J. F. Fraumeni. Jr, "Cancer epidemiology and prevention, third edition," *American Journal of Epidemiology*, 168(4):469, 2008.

[47] K. Raza, and R. Parveen, "Reconstruction of gene regulatory network of colon cancer using information theoretic approach," *4th International Conference: The Next Generation Information Technology Summit*, 9.06-9.06, 2013.

[48] Z. B. Joseph, "Analyzing time series gene expression data," *Bioinformatics*, 20(16):2493-2503, 2004.

[49] Z. Wang, S. Member, V. Palade, and Y. Xu, "Neuro-fuzzy ensemble approach for microarray cancer gene expression data analysis," *Proceedings of the Second International Symposium on Evolving Fuzzy Systems*, 241-246, 2006.

[50] H. Rohian, A. An, J. Zhao, and J. Huang, "Discovering temporal associations among significant changes in gene expression," *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine*, 419-423, 2009.

[51] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa, "KEGG for representation and analysis of molecular networks involving diseases and drugs," *Nucleic Acids Research*, 38:D355-D360, 2009.

[52] J. Kallrath, "Mixed integer optimization in the chemical process industry experience, potential and perspectives," *Chemical Engineering Research and Design*, 78(6):809-822, 2000.

[53] O. Pernia, and J. Scattergood, "What optimisation means for terminal and ports," *Port Technology International*, 59:36-38, 2013.

[54] J. N. Hooker, "Logic, optimization and constraint programming," *INFORMS Journal on Computing*, 14(4):295-321, 2002.

[55] C. Cartis, "C12.1b: Continuous Optimization," *University of Oxford*, 2014. https://www.maths.ox.ac.uk/courses/course/22973.

[56] R. Barták, M. A. Salido, and F. Rossi, "New trends in constraint satisfaction, planning, and scheduling: a survey," *The Knowledge Engineering Review*, 25(3):249-279, 2010.

[57] A. Souza, "Combinatorial algorithms," *Lecture Notes, Winter Term 2010/2011*, 2010.

http://www2.informatik.hu-berlin.de/alcox/lehre/lvws1011/coalg/combinatorial_algorithms.pdf.

[58] R. Bellman, "The theory of dynamic programming," *Bulletin of The American Mathematical Society*, 60(6):503-515, 1954.

[59] J. Erickson, "Dynamic programming," *Lecture Note, Lecture 5*, 2013. http://web.engr.illinois.edu/ jeffe/teaching/algorithms/.

[60] M. Muriati, and A. K. A. Zuriyati, "Protein sequence alignment using dynamic programming," *Australian Journal of Basic and Applied Sciences*, 8(4):47-51, 2014.

[61] http://courses.csail.mit.edu/6.006/fall11/rec/rec21_knapsack.pdf, 2011.

[62] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms, third edition," *The Massachusetts Institute of Technology press*, USA, 2009.

[63] Y. Murasawa, 2012. blade24.eco.osakafu-u.ac.jp/ murasawa/ue-ln05.pdf.

[64] J. Trahan, A. Kaw, and K. Martin, "Computational time for finding the inverse of a matrix: LU decomposition vs. naive gaussian elimination," University of South Florida, 2006.

[65] F. Reichenbach, A. Born, D. Timmermann, and R. Bill, "A distributed linear least squares method for precise localization with low complexity in wireless sensor networks," *Proceedings of the 2nd IEEE International Conference on Distributed Computing in Sensor Systems*, 40:514-528, 2006.

[66] G. H. Golub, and C. F. V. Loan, "Matrix Computations," *The Johns Hopkins University Press*, 1996.

[67] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL, a general reverse engineering algorithm for inference of genetic network architectures," *Proceedings of Pacific Symposium on Biocomputing*, 3:18-29, 1998.

[68] T. F. Liu, W. K. Sung, and A. Mittal, "Learning gene network using time-delayed Bayesian network," *International Journal on Artificial Intelligence Tools*, 15(3):353-370, 2006.

[69] H. Toh, and K. Horimoto, "Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling," *Bioinformatics*, 18(2):287-297, 2002.

[70] R. R. Opgen, and K. Strimmer, "Inferring gene dependency networks from genomic longitudinal data: a functional data approach," *REVSTAT*, 4(1):53-65, 2006.

[71] R. R. Opgen, and K. Strimmer, "From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data," *BMC Systems Biology*, 1:37, 2007.

[72] S. Kimura, S. Nakayama, and M. Hatakeyama, "Genetic network inference as a series of discrimination tasks," *Bioinformatics*, 25(7):918-925, 2009.

[73] S. A. Kauffman, " The origins of order: Self-organization and selection in evolution", *Oxford University Press*, NY, USA, 1993.

[74] T. Tamura, Y. Yamanishi, M. Tanabe, S. Goto, M. Kanehisa, K. Horimoto, and T. Akutsu, "Integer programming-based method for completing signaling pathways and its application to analysis of colorectal cancer," *Genome Informatics*, 24:193-203, 2010.

[75] S. Kim, H. Li, E. R. Dougherty, N. Cao, Y. Chen, M. Bittner, and E. Suh, "Can Markov chain models mimic biological regulations?" *Journal of Biological Systems*, 10(4):337357, 2002.

[76] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Molecular Biology of the Cell*, 9(12):3273-3297, 1998.

[77] J. Khan, N. Bouaynaya, and H. M. Fathallah-Shaykh, "Tracking of time-varying genomic regulatory networks with a LASSO-Kalman smoother," *EURASIP Journal on Bioinformatics and Systems Biology*, 2014(1):3, 2014.

[78] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C. H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C. T. Brown, C. B. Livi, P. Y. Lee, R. Revilla, A. G. Rust, Z. J. Pan, M. J. Schilstra, P. J. Clarke, M. I. Arnone, L. Rowen, R. A. Cameron, D. R. McClay, L. Hood, and H. Bolouri, "A genomic regulatory network for development," *Science*, 295(5560):1669-1678, 2002.

[79] N. M. Luscombe, M. M. Babu, H. Yu, M. Snyder, S. A. Teichmann, and M. Gerstein, "Genomic analysis of regulatory network dynamics reveals large topological changes," *Letters to Nature*, 431:308-312, 2004.

[80] K. H. Cho, S. M. Choo, S. H. Jung, J. R. Kim, H. S. Choi, and J. Kim, "Reverse engineering of gene regulatory networks," *IET Systems Biology*, 1(3):149-163, 2007.

[81] H. Hache, H. Lehrach, and R. Herwig, "Reverse engineering of gene regulatory networks: a comparative study," *EURASIP Journal on Bioinformatics and Systems Biology*, 2009, 617281, 2009.

[82] G. Rassol, and N. Bouaynaya, "Inference of time-varying gene networks using constrained and smoothed Kalman filtering," *IEEE International Workshop on Genomic Signal Processing and Statistics*, 172-175, 2012.

[83] J. Cao, X. Qi, and H. Zhao, "Modeling gene regulation networks using ordinary differential equations," *Methods in Molecular Biology*, 802:185-197, 2012.

[84] R. Yoshida, S. Imoto, and T. Higuchi, "Estimating time-dependent gene networks from time series microarray data by dynamic linear models with Markov switching," *Proceedings of IEEE Computational Systems Biology*, 289-298, 2005.

[85] A. Fujita, J. R. Sato, H. M. Garay-Malpartida, P. A. Morettin, M. C. Sogayar, and C. E. Ferreira, "Time-varying modeling of gene expression regulatory networks using the wavelet dynamic vector autoregressive method," *Bioinformatics*, 23(13):1623-1630, 2007.

[86] J. W. Robinson, and A. J. Hartemink, "Non-stationary dynamic Bayesian networks," *Advances in Neural Information Processing Systems*, 1369-1376, 2008.

[87] S. Lèbrel, J. Becq, F. Devaux, M. P. H. Stumpf, and G. Lelandais, "Statistical inference of the time-varying structure of gene-regulation networks," *BMC Systems Biology*, 4(130), 2010.

[88] T. Thorne, and M. P. H. Stumpf, "Inference of temporally varying Bayesian Networks," *Bioinformatics*, 28(24):3298-3305, 2012.

[89] Y. W. Teh, and M. I. Jordan, "Hierarchical Bayesian nonparametric models with applications," *Bayesian Nonparametrics, Cambridge University Press*, Cambridge, UK, 158-207, 2010.

[90] A. Ahmed, L. Song, and E. P. Xing, "Time-varying networks: recovering temporally rewiring genetic networks during the life cycle of drosophila," *SCS Technical Report Collection*, CMU-ML-08-118, 2008.

[91] N. Nakajima, and T. Akutsu, "Network completion for time-varying genetic networks," *Proceedings of the 7th International Conference on Complex, Intelligent, and Software Intensive Systems*, 2013, 553-558, 2013.

[92] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, 453:98-101, 2008.

[93] N. Nakajima, T. Tamura, Y. Yamanishi, K. Horimoto, and T. Akutsu, "Network completion using dynamic programming and least-squares fitting," *The Scientific World Journal*, 2012:957620, 2012.

[94] M. N. Arbeitman, E. E. Furlong, F. Imam, E. Johnson, B. H. Null, B. S. Baker, M. A. Krasnow, M. P. Scott, R. W. Davis, and K. P. White, "Gene expression during the life cycle of Drosophila melanogaster." *Science*, 297(5590):2270-2275, 2002.

[95] N. Noman, L. Palafox, and H, Iba, "On model selection criteria in reverse engineering gene networks using RNN model," *Proceedings of International Conference on Convergence and Hybrid Information Technology*, (1)2012:155-164, 2012.

[96] T. Schaffter, D. Marbach, and D. Floreano, "GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods," *BMC Bioinformatics*, 27(16):2263-2270, 2011.

[97] I. M. Bocicor, G. Caravagna, A. Graudenzi, C. Cava, G. Mauri, and M. Antoniotti, "Ordering copy number alteration data to analyze colorectal cancer progression," *Network Tools and Applications in Biology 2013*, 3(18):1569, 2013.

[98] J. R. Nevins, "The Rb/E2F pathway and cancer," *Human Molecular Genetics*, 10(7):699-703, 2001.

[99] P. O. Humbert, R. Verona, J. M. Trimarchi, C. Rogers, S. Dandapani, and J. A. Lees, "E2f3 is critical for normal cellular proliferation," *Genes and Development*, 14(6):690-703, 2000.

[100] J. M. Trimarchi, and J. A. Lees, "Sibling rivalry in the E2F family," *Nature Reviews Molecular Cell Biology*, 3(1):11-20, 2002.

[101] D. G. Beer, S. L. R. Kardia, C. C. Huang, T. J. Giordano, A. M. Levin, D. E. Misek, L. Lin, G. Chen, T. G. Gharib, D. G. Thomas, M. L. Lizyness, R. Kuick, S. Hayasaka, J. M. G. Taylor, M. D. Iannettoni, M. B. Orringer, and S. Hanash, "Gene-expression profiles predict survival of patients with lung adenocarcinoma," *Nature Medicine*, 8(8):816-824, 2002.

[102] Y. J. Choi, and C. Kendziorski, "Statistical methods for gene set co-expression analysis," *Bioinformatics*, 25(21):2780-2786, 2009.

[103] J. P. Alao, "The regulation of cyclin D1 degradation: roles in cancer development and the potential for therapeutic invention," *Molecular Cancer*, 6(24), 2007.

List of Publications

Journal papers

- N. Nakajima, T. Tamura, Y. Yamanishi, K. Horimoto, and T. Akutsu, "Network completion using dynamic programming and least-squares fitting," *The Scientific World Journal*, vol. 2012, Article ID 957620, 8 pages, 2012.

- N. Nakajima, and T. Akutsu, "Exact and heuristic methods for network completion for time-varying genetic networks," *BioMed Research International*, vol. 2014, Article ID 684014, 14 pages, 2014.

- N. Nakajima, and T. Akutsu, "Network completion for static gene expression data," *Advances in Bioinformatics*, vol. 2014, Article ID 382452, 9 pages, 2014.

Conference paper

- N. Nakajima, and T. Akutsu, "Network completion for time-varying genetic networks," *Proceedings of the 7th International Conference on Complex, Intelligent, and Software Intensive Systems*, vol. 2013, pp. 553-558, 2013.